Introduction to statistics / statistical tools & the HistFitter framework

Max Baak (CERN), Geert-Jan Besjes (Nijmegen/Nikhef), Jeanette Lorenz (LMU/Excellence Cluster Universe), Sophio Pataraia (Bergische Universitaet Wuppertal)

+ many other people

15 April 2015

ATLAS EXPERIMENT
http://atlas.ch

HistFitter

Overview

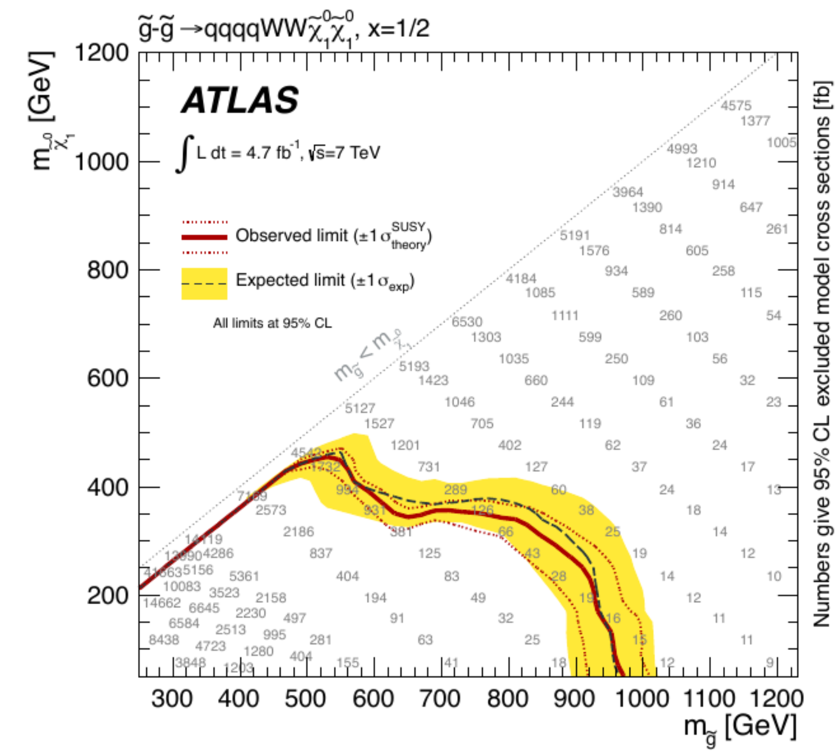
- Introduction to statistics (short)
- Introduction to statistical analysis (RooFit, RooStats, HistFactory)
- HistFitter overview
  - Introduction & strategy
- HistFitter tutorial
  - Running a fit & visualization
  - Calculating limits

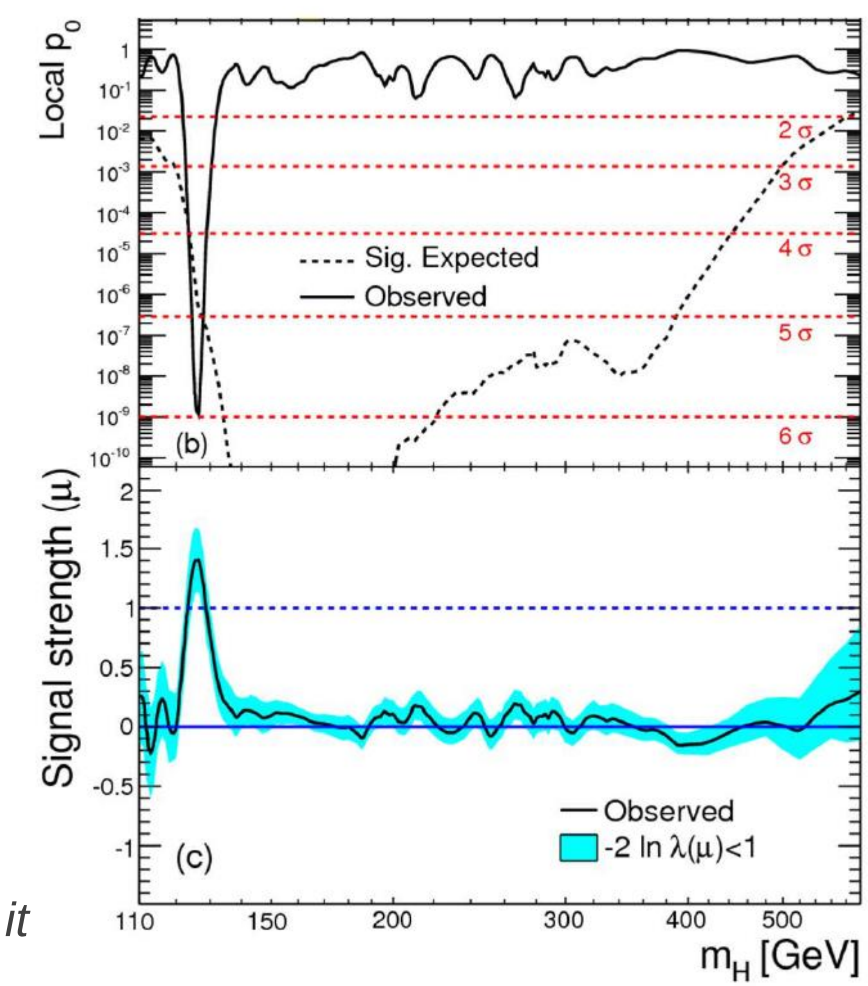
Introduction to HEP statistics

Largely borrowed from lectures/slides by W. Verkerke

Basic questions

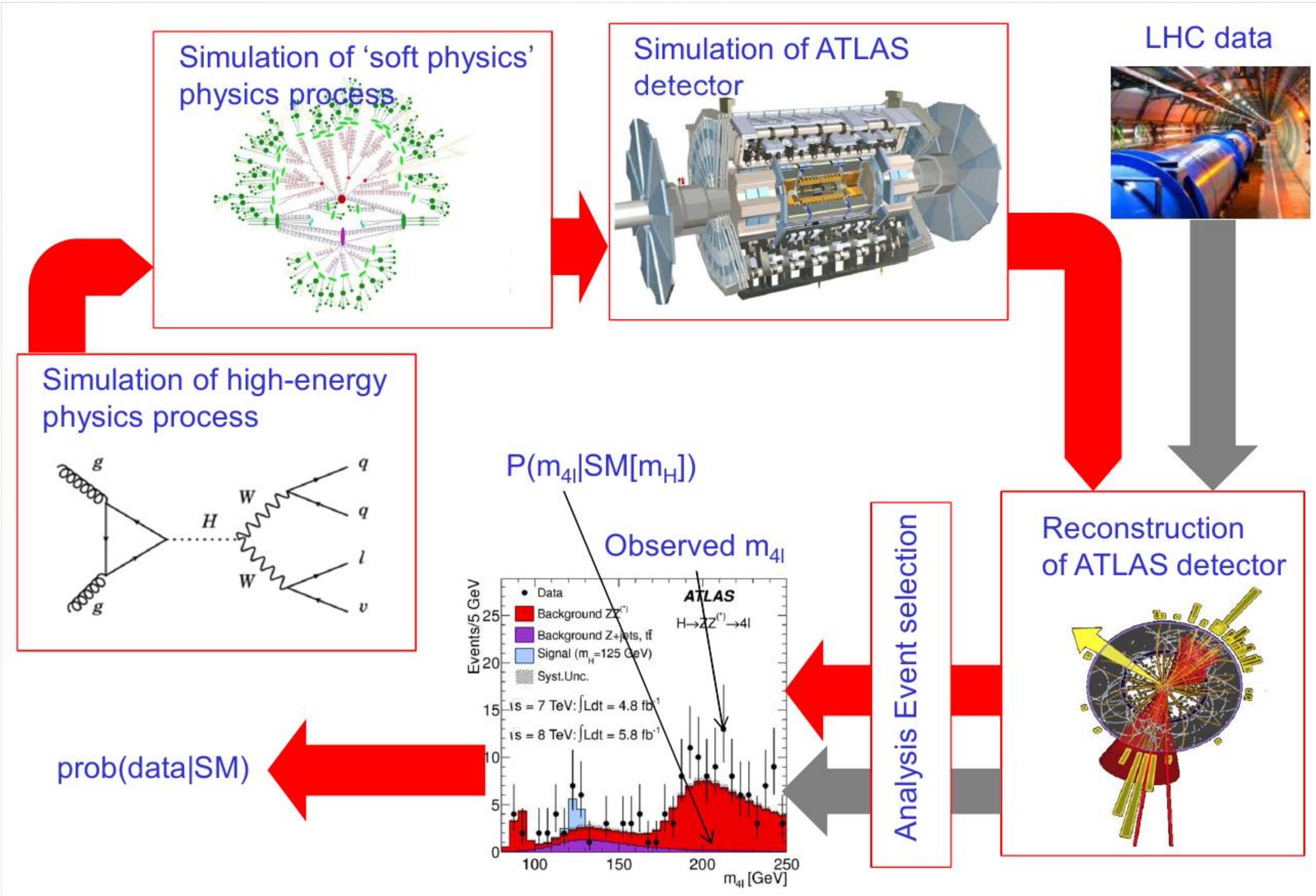
- Physics questions we want to answer...
  - Is the new discovered particle a ‘vanilla’ Higgs boson?
  - What is its production cross section and couplings?
  - Is there any SUSY in ATLAS data?
    - If not, what models do not agree with data?
- Enormous efforts in many channels, millions of plots with signal/backgrounds expectations, with systematics and observed data
- How do you conclude on these questions?
- Statistical tests construct probabilistic statements/models on P(theory|data) or P(data|theory)
  - Likelihood fits
  - Systematics/uncertainties
  - Hypothesis testing
  - Setting limits ...
- Result: decisions based on these tests





As a layman I would now say, I think we have it

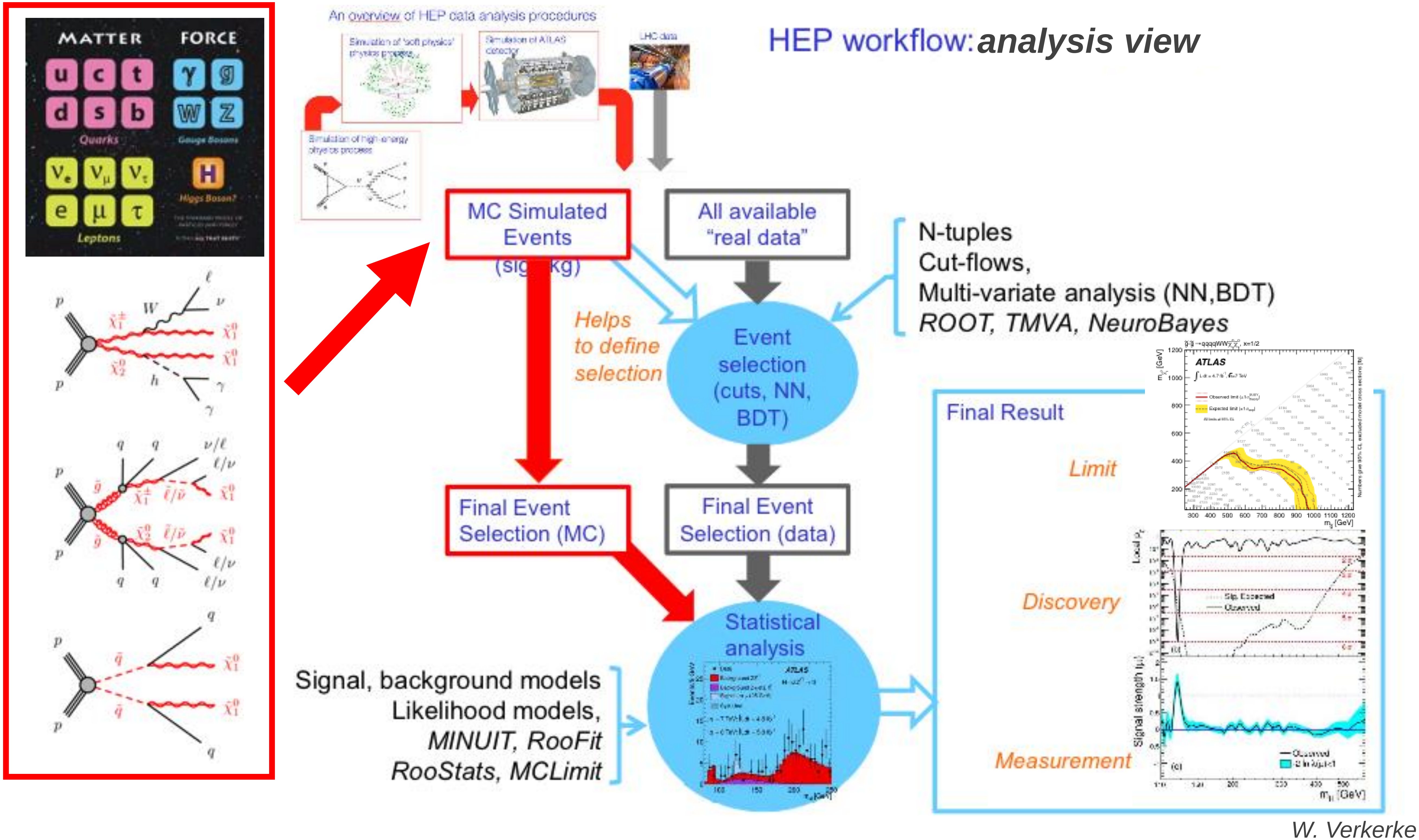
HEP workflow



W. Verkerke

- If you paid attention in xAOD tutorial, this should be a piece of cake...

HEP data analysis



- HEP Data Analysis is (should be) for a large part the reduction of a physics theory(s) to a statistical model
- Statistical/probability model: Given a measurement ***x*** (eg N events), what is the probability to observe each possible ***x***, under the hypothesis that the physics theory is true?

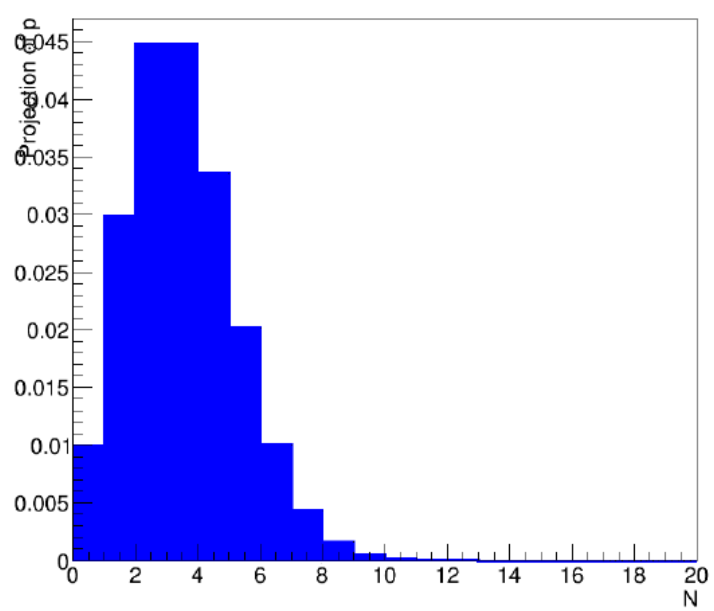
Simple statistical example

- Central concept in statistics is the ‘probability model’ : assigns a probability to each possible experimental outcome

- <u>Example</u>: a HEP counting experiment
  - Count number of events in your signal region (SR) in your data (specific lumi): Poisson distribution $P(N\,|\,\mu) = \dfrac{\mu^{N} e^{-\mu}}{N!}$
  - Given the *expected(MC)* event count, the probability model is fully specified

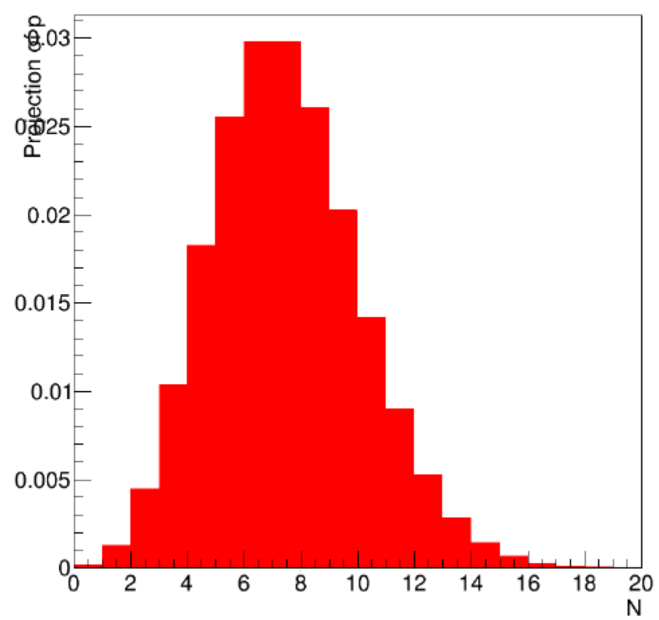
PROBABILITY

μ=3 (“bkg only”)

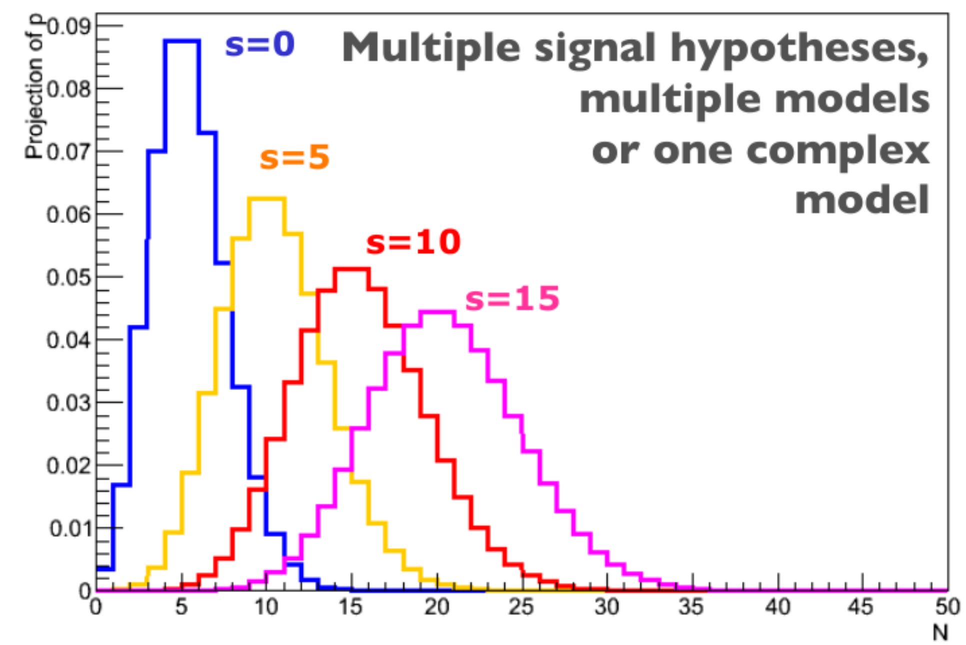


Poisson(N| b)

μ=7 (“bkg+signal”)



Poisson(N| s + b)



Poisson(N| s + b)

- Suppose we measure N = 7 events (Nobs), then can calculate the probability
- P(Nobs|hypothesis) is called **<u>LIKELIHOOD</u>** **- L(Nobs|b), L(Nobs|s+b), L(observed data|theory)**

p(Nobs|b) = 2.2% **p(Nobs|s+b) = 14.9%**

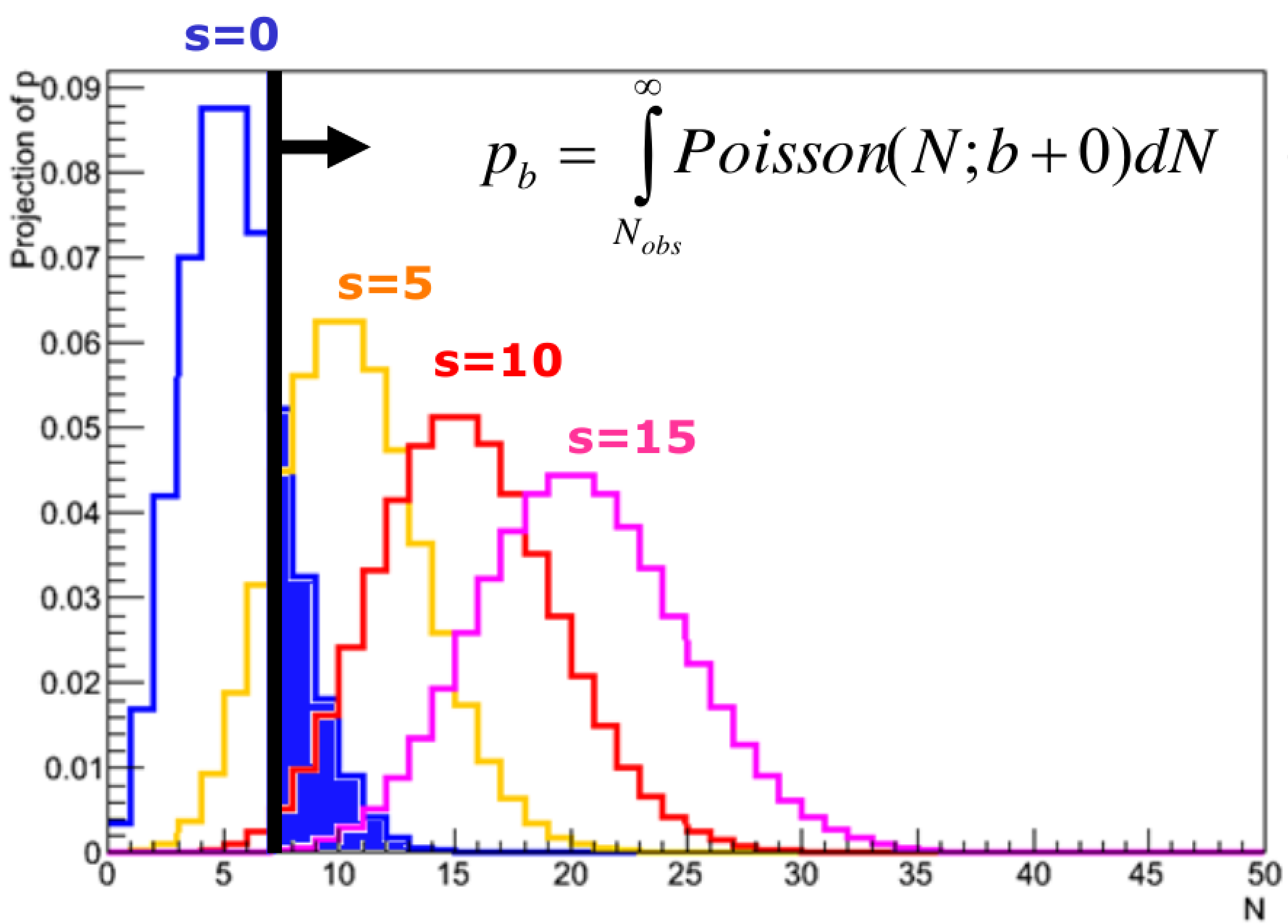
  - Data is more likely under s+b hypothesis than bkg-only

W. Verkerke

p-value

- **P-VALUE:** probability to obtain observed data, or more extreme, given the hypothesis in future repeated identical experiments

- For our example from previous page:
  - For the bkg-only hypothesis: $\mathbf{p_b}$ = Fraction of future measurements with N=Nobs (or larger) if s=0



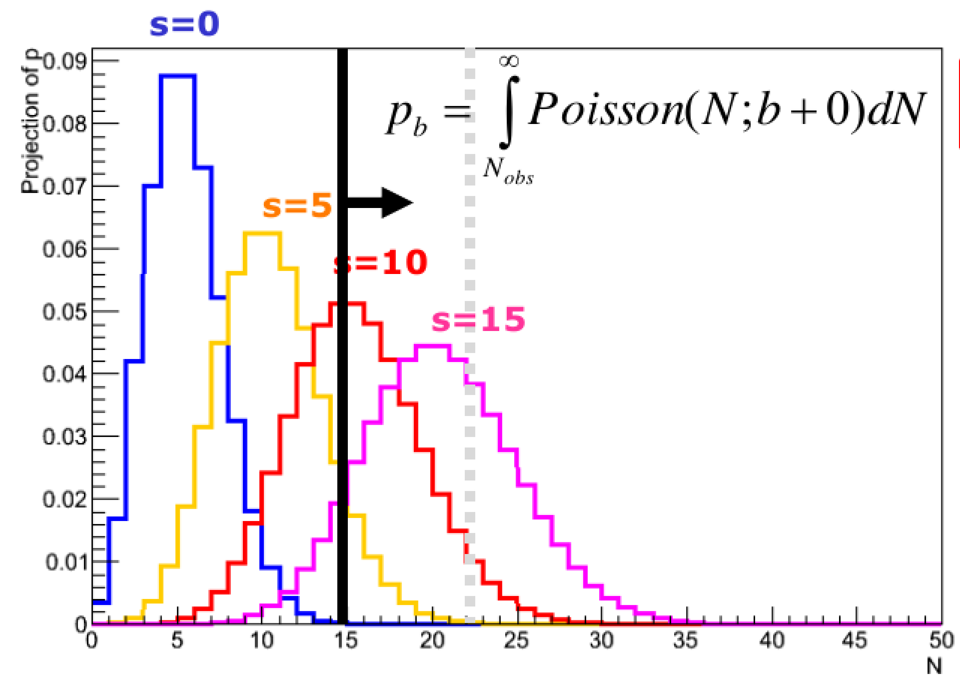
$$p_b = \int_{N_{obs}}^{\infty} Poisson(N; b+0)\,dN \quad (=0.23)$$

- Frequentist p-values (apologies to Bayesians) -- see links later

Excess over background

- p_b or p-values of background hypothesis is used to quantify ‘discovery’
- ‘discovery’ = excess of events over background expectation

- One more example:
  - Nobs=15 for same model, what is p_b?



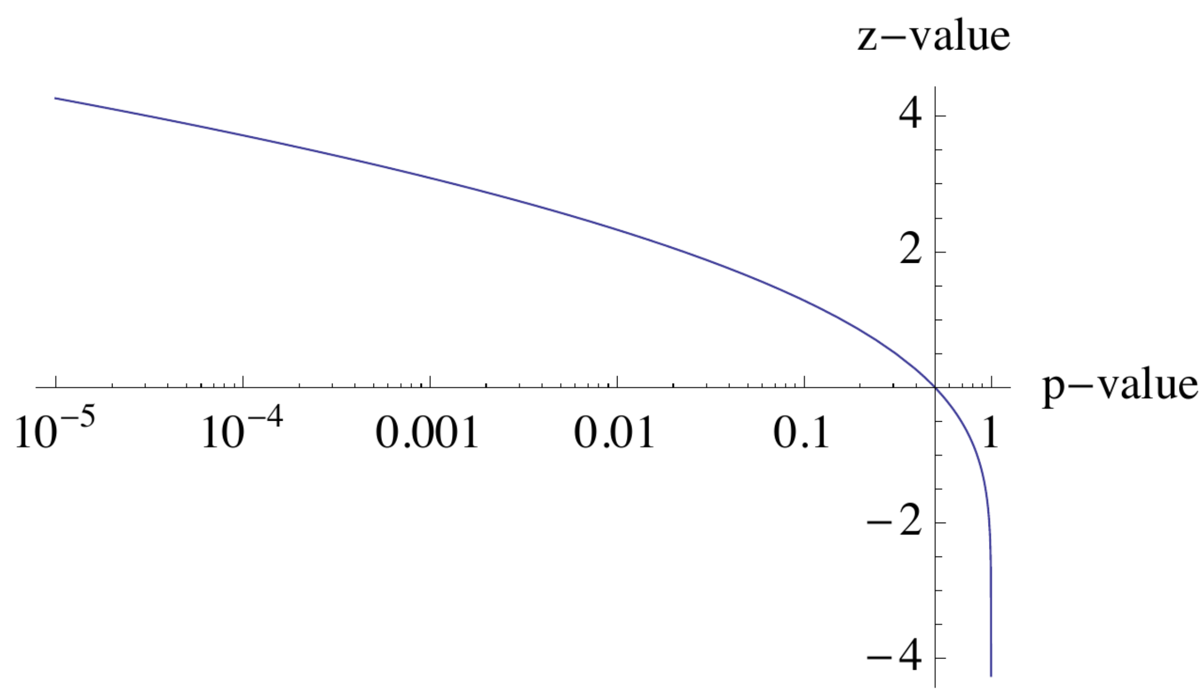
$$p_b = \int_{N_{obs}}^{\infty} Poisson(N; b+0)\,dN$$

$\boxed{(=0.00022)}$

- Results customarily expressed as odds of a Gaussian fluctuation with equal p-value: **significance, Zn, z-value**

- Nobs = 15 → Zn = 3.5σ

- Nobs = 22 → **Zn = 5σ**
  or **p_b < 2.87 ×10^{-7}**



z-value =

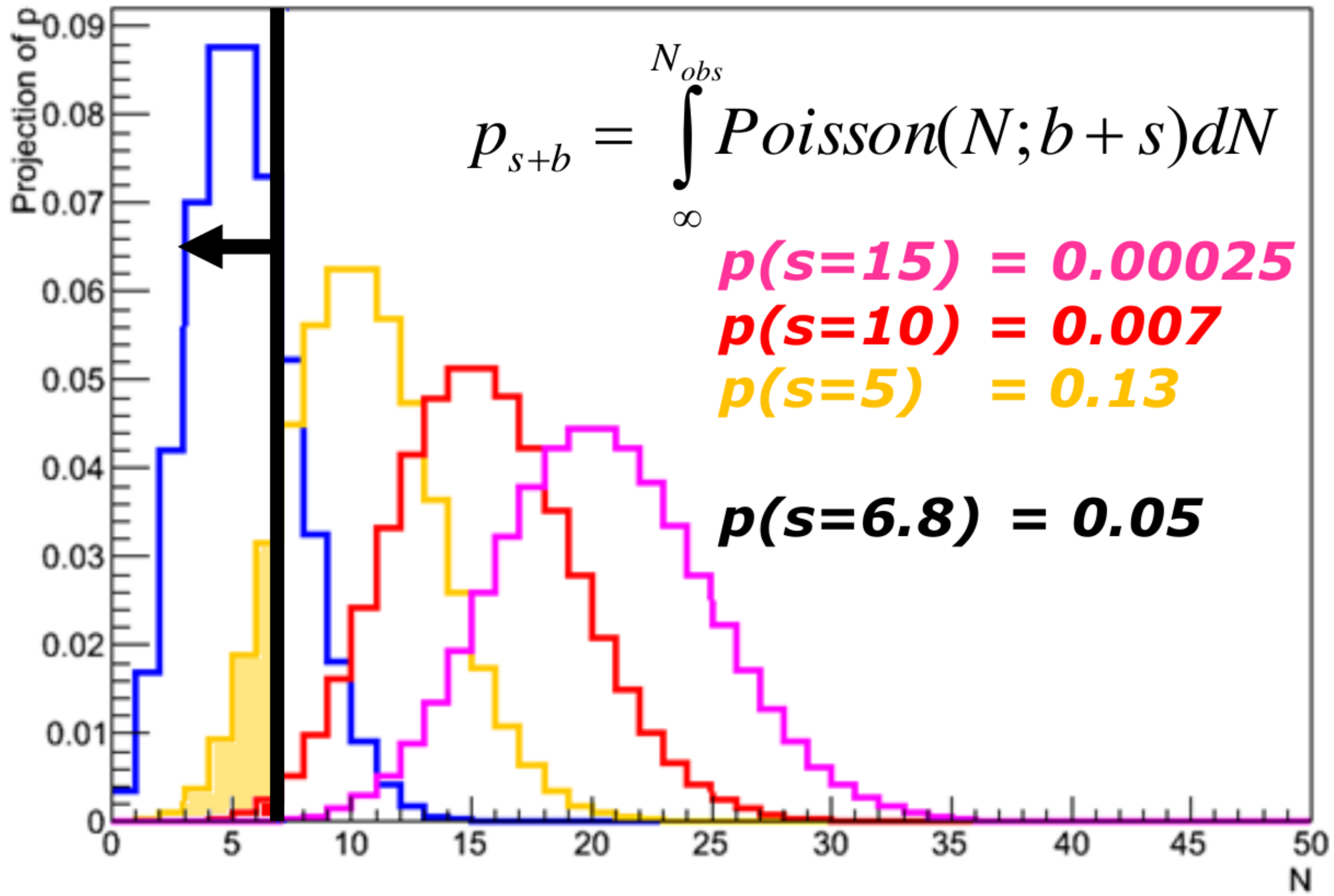
$\texttt{sqrt(2.)} * \texttt{TMath} :: \texttt{ErfInverse}(1. - 2. * \texttt{pvalue})$

$$p\text{-value} = \int_{z\text{-value}}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}\,dx,$$

Fig. 1. Relationship between p-value and z-value.

Upper limits

- Can also define p-value for s+b hypothesis $\mathbf{p_{s+b}}$
  - Note convention change: integration range in $\mathbf{p_{s+b}}$ is flipped



- Convention: express result as value (upper limit) of **s** for which $\mathbf{p_{s+b}} = 5\%$

  or excluded at 95% confidence level (95% C.L.)
- Our example:
  - s>6.8 is excluded at 95% C.L.

Modified Upper limits : CLs

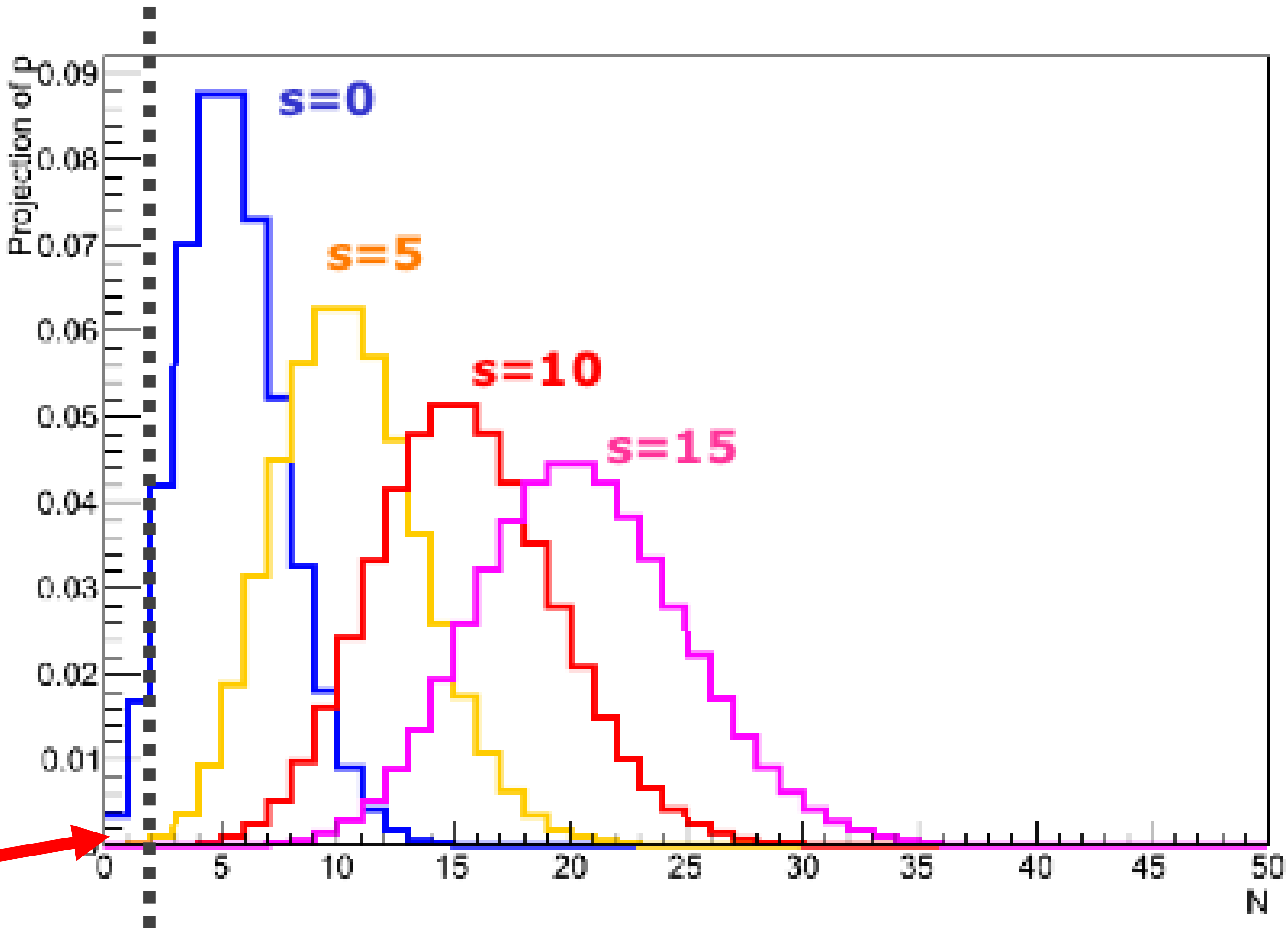
- Interpretation of $\mathbf{p_{s+b}}$ in terms of inference on signal only is problematic
  - Since $\mathbf{p_{s+b}}$ quantifies consistency with data of signal + background
  - Problem apparent when observed data has downward fluctuation wrt background expectation

- Example: Nobs = 2 → $\mathbf{p_{s+b}(s=0) = 0.04}$
  - **s≥0 excluded at 95% C.L. ???**

- Modified approach to protect against such inference on signal (LHC convention):
  - Instead of requiring $\mathbf{p_{s+b}}$ = 5%,

    require

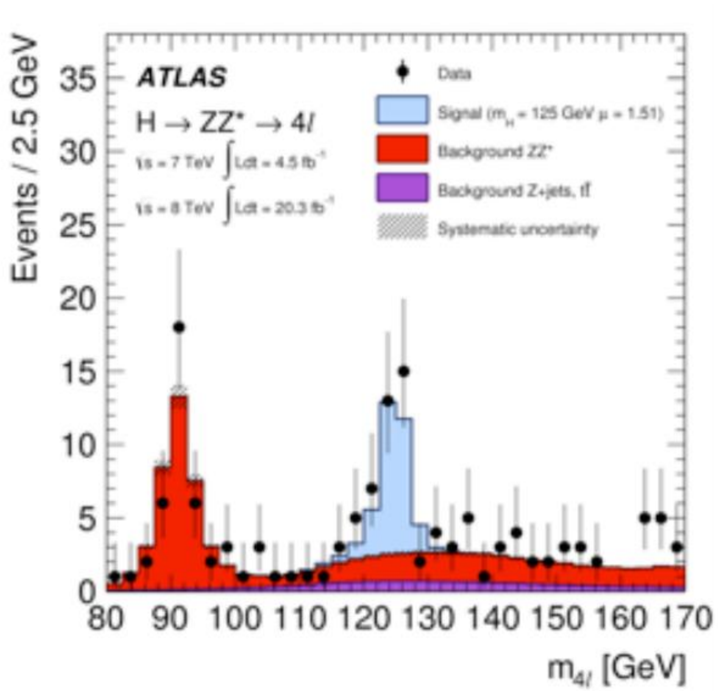
$$\mathrm{CL}_s \equiv \frac{p_{s+b}}{\boxed{1 - p_b}} = 5\%$$

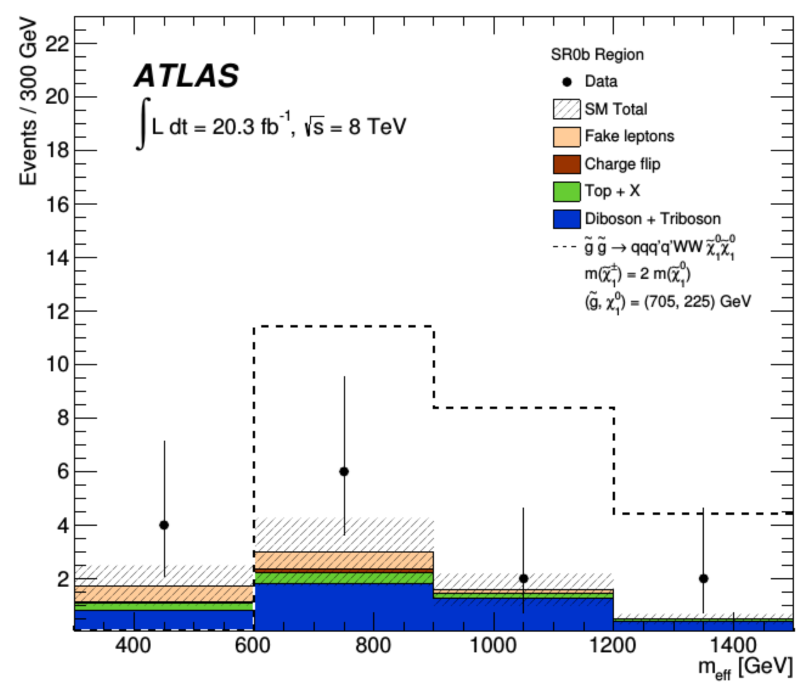


- Example: Nobs = 2 → **s>3.4 excluded at 95% CLs**
- For large Nobs effect on limit is small as $\mathbf{p_b} \to 0$
- https://twiki.cern.ch/twiki/pub/AtlasProtected/StatisticsTools/CLsInfo.pdf

More complex examples

- Typical analysis is not a simple counting experiment
  - Many intrinsic uncertainties on signal and bkg
  - Result is a distribution, not a single number
    - SUSY searches: discovery is cut&count, but many exclusion limits are shape-fits/multi-bin





- Any result can be converted into a single number by constructing a ***test statistic***
  - A test statistic compresses all signal-to-background discrimination power into one number
  - Most powerful discriminators are ***Likelihood Ratios*** *(Neyman-Pearson lemma)*
  - q_μ is a common test statistic (LHC convention)

$$q_\mu = -2\ln\frac{L(data\,|\,\mu)}{L(data\,|\,\hat{\mu})}$$

Likelihood ratio test statistic

- **Signal strength μ** = signal rate / nominal signal rate (also know as μ_{SIG})
  - Bkg-only hypothesis: **μ** = 0
  - Bkg + signal hypo: **μ** = 1
  - Bkg + 2 X signal hypo: **μ** = 2
- Likelihood with nominal signal strength (**μ** = 1)

'likelihood assuming nominal signal strength'

$$q_1 = -2\ln\frac{L(data\,|\,\mu=1)}{L(data\,|\,\hat{\mu})}$$

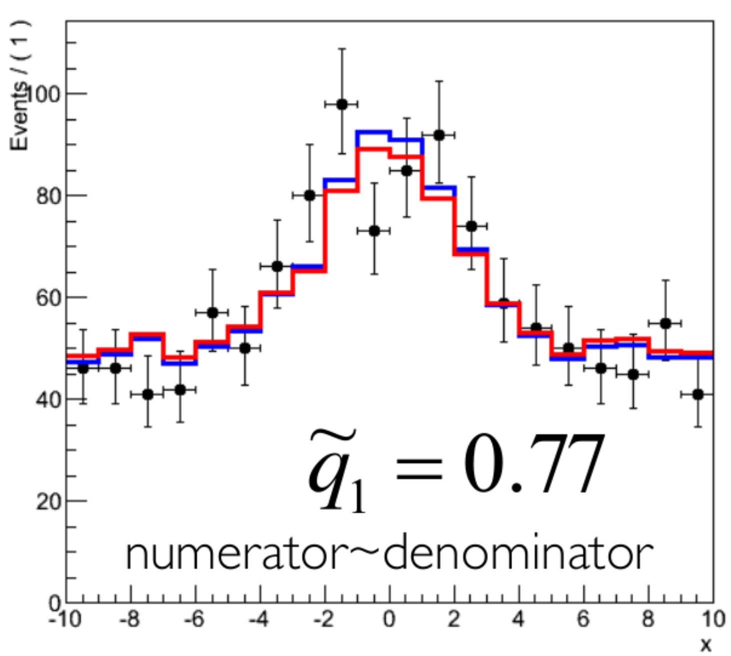
$\hat{\mu}$ is best fit value of μ

'likelihood of best fit'

- **Example:** simple s + b model with no uncertainties

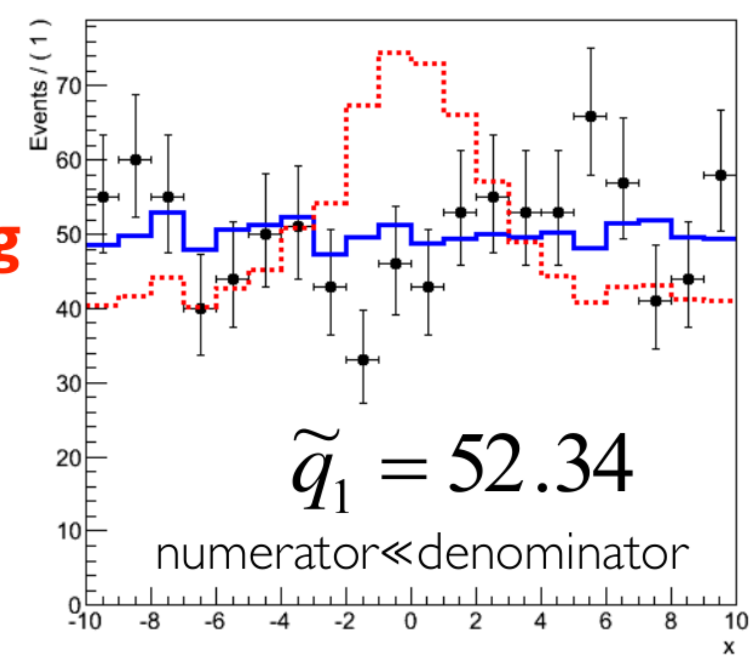
*On signal-like data q1 is **small***

*On background-like data q1 is **large***



signal(μ=1)+bkg

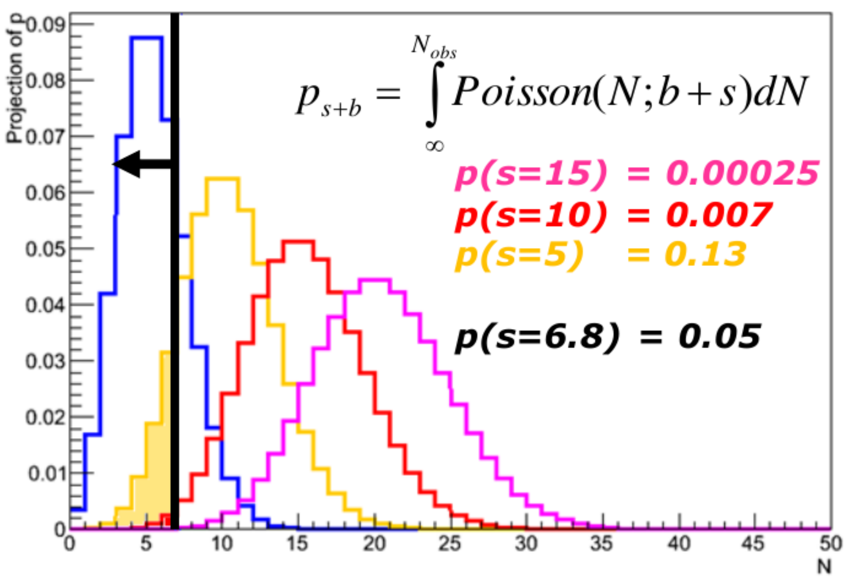
best fit

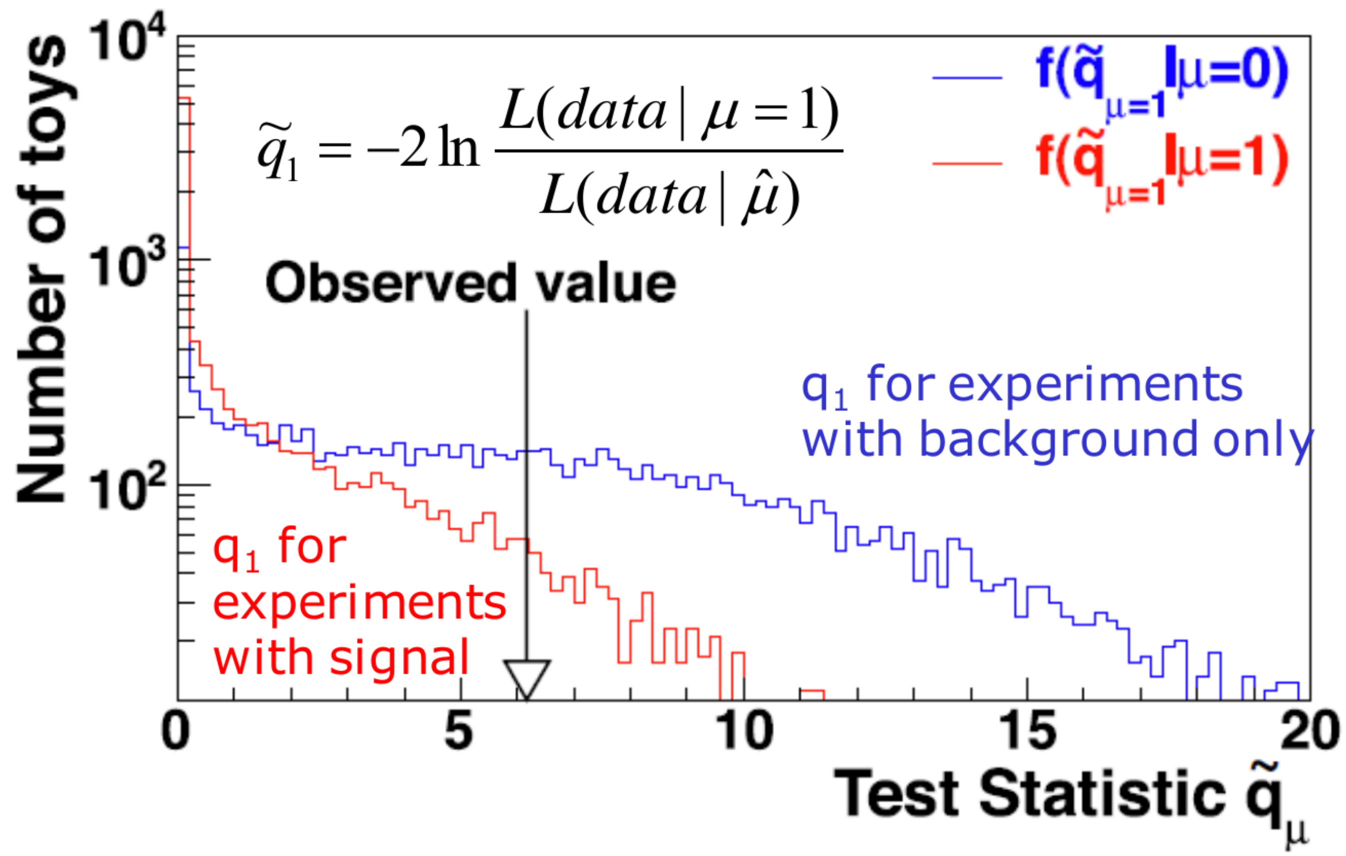


Distribution of test statistic

- Value of q_1 on data is now the *measurement*
- Distribution of q_1 is **not** calculable → But can be obtained from pseudo-experiments (toys)
  - Generate a large number of pseudo-experiments with a given value of **μ**, calculate q_1 for each, plot distribution

Note analogy to Poisson counting example





- From q_{obs} and these test statistic distributions, f(q_μ), can then set limits or calculate discovery significance similar to what was shown for Poisson example

- Typically CPU-intensive to run many toy-experiments → approximate with **asymptotic formulae**, aka **asimov data** (only works in cases when Nobs≳10, see links for details)

Systematic uncertainties

- Typically HEP models will have uncertainties: experimental (JES,trigger eff.) or theoretical (Q,σ)

$$L(data\,|\,\mu) \rightarrow L(data\,|\,\mu,\vec{\theta})$$

$$L(data\,|\,\mu,\theta) = Poisson(N_i\,|\,\mu\cdot s_i(\theta)+b_i(\theta))\cdot p(\tilde{\theta},\theta)$$

- Models w/ uncertainties, described by additional parameters θ that describe effect of uncert.
- Likelihood includes *auxiliary measurement* terms that constrain the nuisance parameters θ
  - Auxiliary measurement given by performance group (jet perf.) or theory variations (renorm. scale up/down)

- Likewise uncertainties quantified by nuisance parameters are incorporated into test statistic using **Profile Likelihood Ratio**

$$q_\mu = -2\ln\frac{L(data\,|\,\mu)}{L(data\,|\,\hat{\mu})} \longrightarrow \tilde{q}_\mu = -2\ln\frac{L(data\,|\,\mu,\hat{\hat{\theta}}_\mu)}{L(data\,|\,\hat{\mu},\hat{\theta})}$$

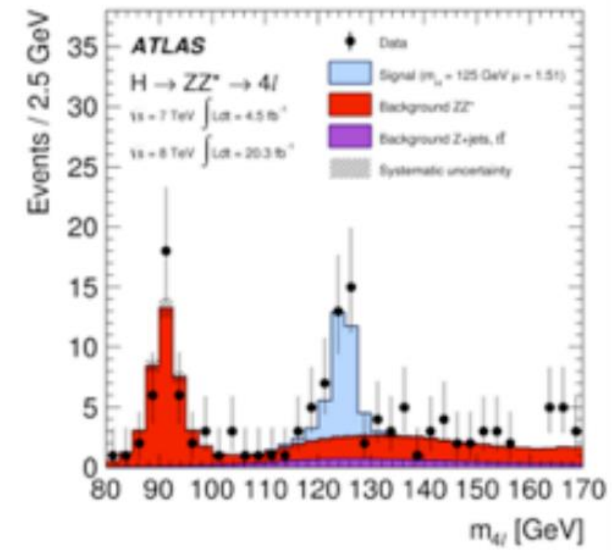
'likelihood of best fit for a **given fixed value of μ'**

'likelihood of best fit'

(with a constraint $0 \leq \hat{\mu} \leq \mu$)

Overview for a search

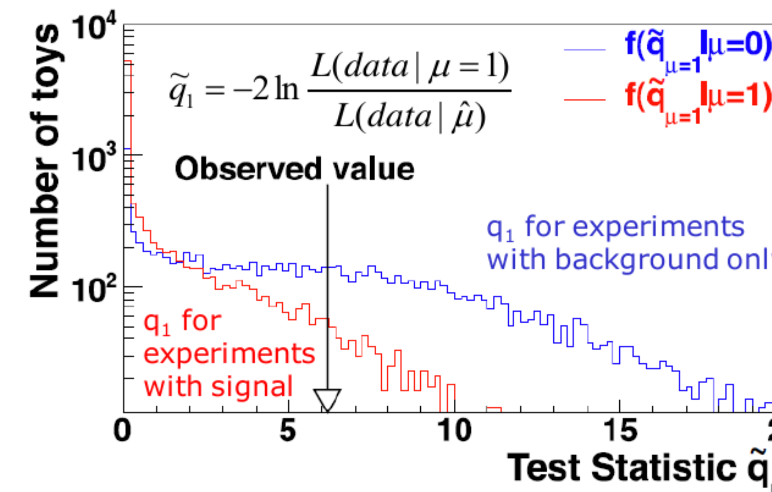
- Take Higgs search as example, and put it all together



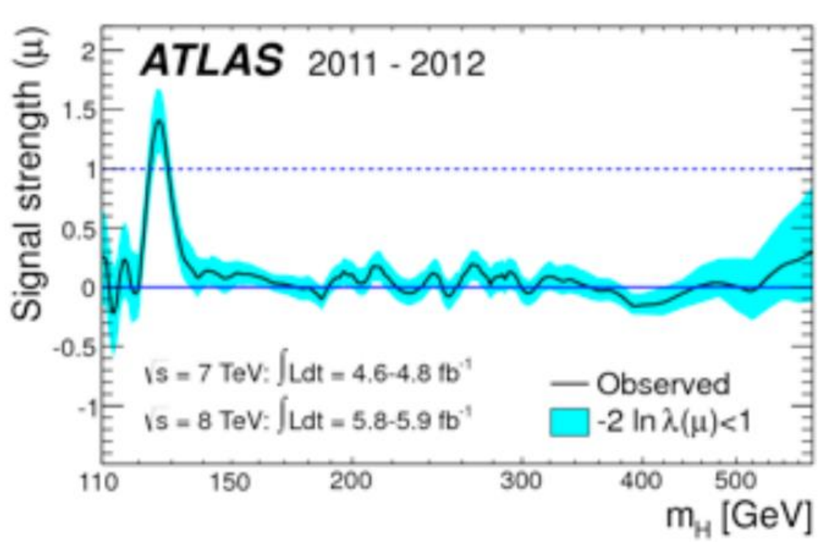
- Result from data is a distribution (eg m(4l))
- Model signal and background by PDF (probability density function) for a given Higgs mass hypothesis

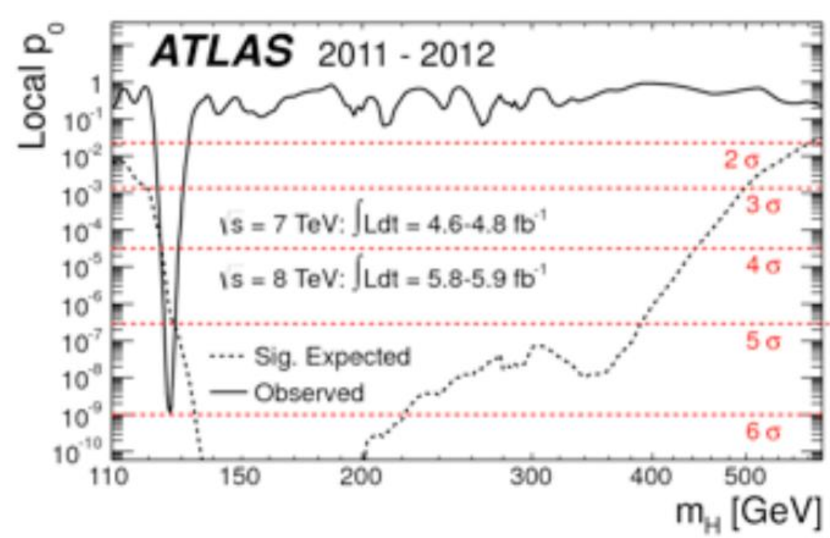
- Construct likelihood(s) by joining data and model(s)
- Construct test statistic q_μ from likelihoods

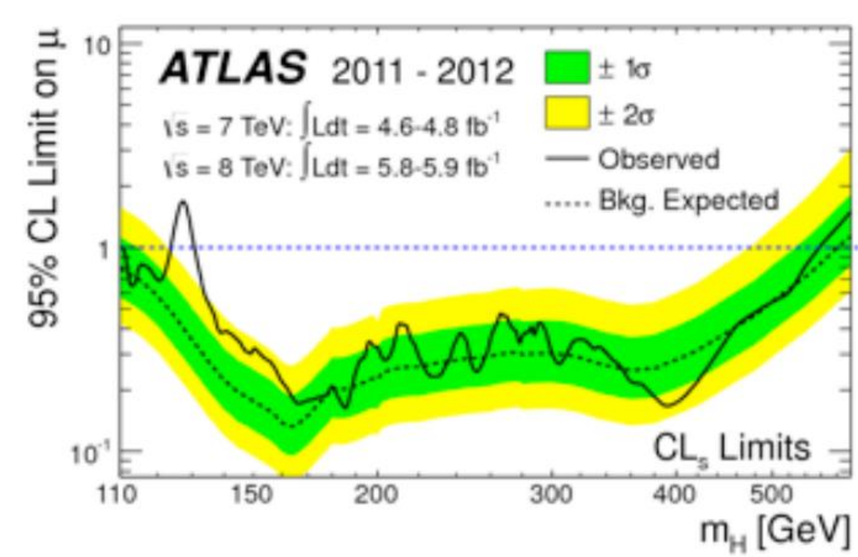
$$\tilde{q}_\mu(m_H) = -2\ln\frac{L(data\,|\,\mu, m_H, \hat{\hat{\theta}}_\mu)}{L(data\,|\,\hat{\mu}, m_H, \hat{\theta})}$$



- Obtain expected distributions of q_μ
- Determine discovery p_0 and signal exclusion limit
- Repeat for each assumed m_H







Links

- Statistics lectures (CERN school, 2014, W. Verkerke):
  - Part-1: https://indico.cern.ch/event/287744/contribution/7/material/slides/0.pdf
  - Part-2: https://indico.cern.ch/event/287744/contribution/11/material/slides/1.pdf
  - Part-3: https://indico.cern.ch/event/287744/contribution/14/material/slides/0.pdf

- Plotting the Differences Between Data and Expectation, G. Choudalakis, D. Casadei http://arxiv.org/abs/1111.2062

- CLs: https://twiki.cern.ch/twiki/pub/AtlasProtected/StatisticsTools/CLsInfo.pdf

[28] A. Read, Presentation of search results: the CL s technique, Journal of Physics G: Nuclear and Particle Physics 28 (10) (2002) 2693.

[29] G. Cowan, K. Cranmer, E. Gross, O. Vitells, Asymptotic formulae for likelihood-based tests of new physics, Eur.Phys.J. C71 (2011) 1554. arXiv:1007.1727, doi:10.1140/epjc/s10052-011-1554-0.

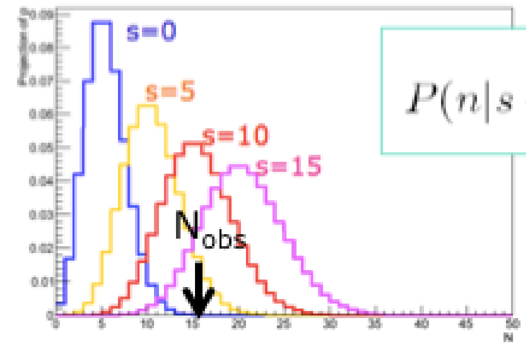
[30] S. Wilks, The large-sample distribution of the likelihood ratio for testing composite hypotheses, Ann. Math. Statist. 9 (1938) 60–62.

Introduction to statistics tools

Largely borrowed from lectures/slides by W. Verkerke

LIKELIHOOD, LIKELIHOOD, LIKELIHOOD...

- **All** fundamental statistical procedures are based on the likelihood function as 'description of the measurement'



$$P(n|s+b) = \frac{(s+b)^n}{n!} e^{-(s+b)}$$

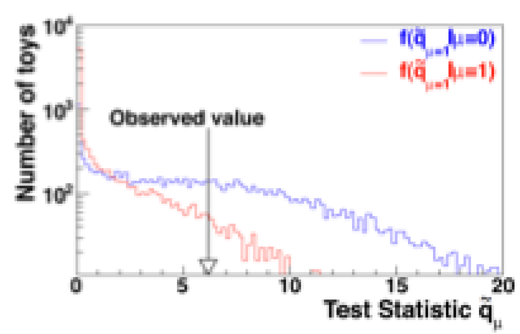
NB: b is a constant in this example

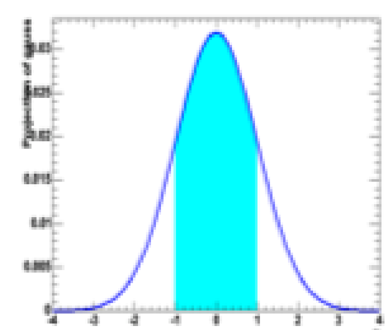
Definition: the Likelihood is P(observed data|theory)

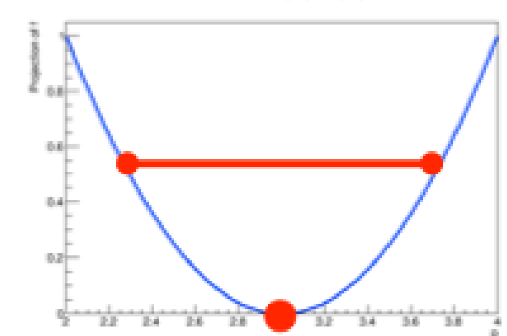
e.g. L(15|s=0)

e.g. L(15|s=10)

Frequentist statistics	Bayesian statistics	Maximum Likelihood
$\lambda_\mu(\vec{N}_{obs}) = \dfrac{L(\vec{N}\,|\,\mu)}{L(\vec{N}\,|\,\hat{\mu})}$	$P(\mu) \propto L(x\,|\,\mu) \cdot \pi(\mu)$	$\left.\dfrac{d \ln L(\vec{p})}{d\vec{p}}\right\rvert_{p_i=\hat{p}_i} = 0$







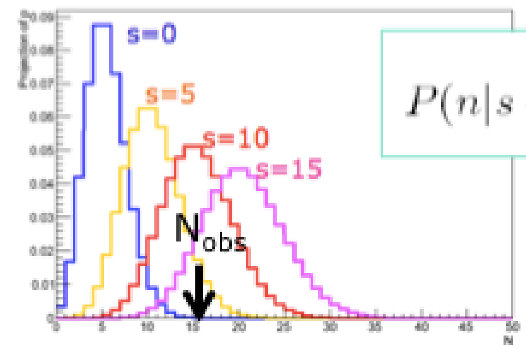
Confidence interval or p-value	**Posterior on s or Bayes factor**	**s = x ± y**

Wouter Verkerke, NIKHEF

Modular software design

- **RooFit:** tool/language for building probability models: datasets, likelihoods, minimization, toy data, visualization
- **HistFactory:** tool to construct binned template models of arbitrary complexity using classes of physics concepts: channel/region, sample, uncertainties
  Builds a RooFit stat. model from HistFactory physics model
- **RooWorkspace:** persistent RooFit object to transport a likelihood, containing model/data. Completely factorizes process of building and using likelihood functions.
- **RooStats:** tool/suite to calculate intervals and perform hypothesis tests using a variety of statistical techniques; easy to use with RooWorkspace

- **All** fundamental statistical procedures are based on the likelihood function as 'description of the measurement'



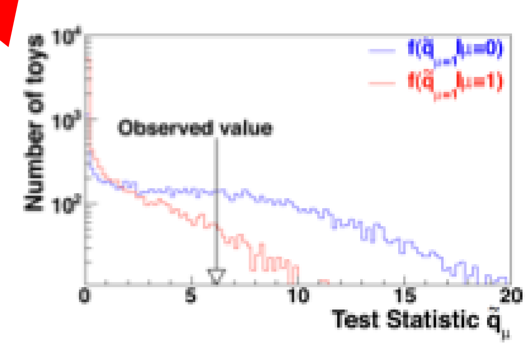
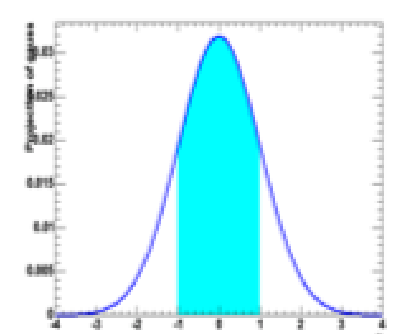
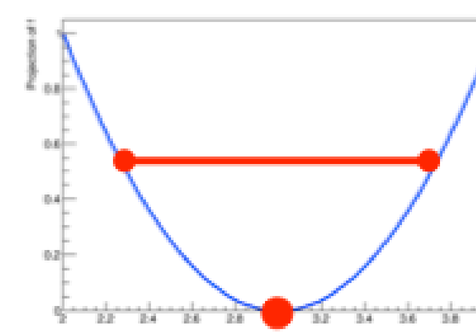
$$P(n|s+b) = \frac{(s+b)^n}{n!} e^{-(s+b)}$$

NB: b is a constant in this example

Definition: the Likelihood is P(observed data|theory)

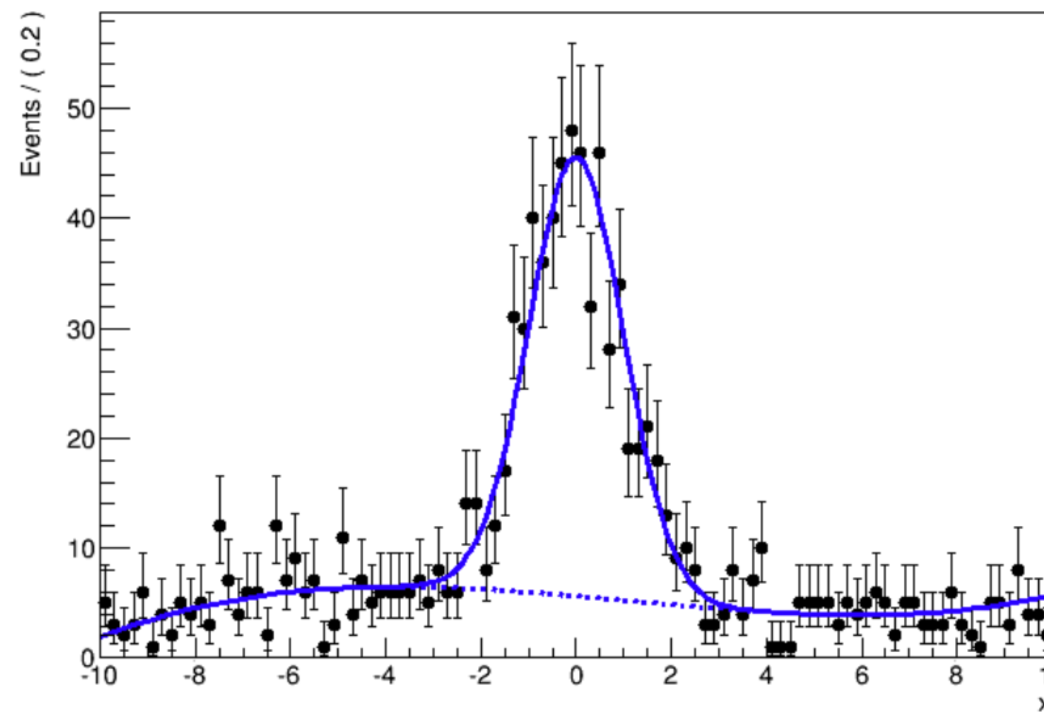
e.g. L(15|s=0)

e.g. L(15|s=10)

Frequentist statistics	Bayesian statistics	Maximum Likelihood
$\lambda_\mu(\vec{N}_{obs}) = \frac{L(\vec{N}\,\vert\,\mu)}{L(\vec{N}\,\vert\,\hat{\mu})}$	$P(\mu) \propto L(x\,\vert\,\mu)\cdot\pi(\mu)$	$\left.\frac{d\ln L(\vec{p})}{d\vec{p}}\right\vert_{p_i=\hat{p}_i} = 0$
		
Confidence interval or p-value	Posterior on s or Bayes factor	s = x ± y

RooFit

- Focus: coding a probability density function PDF : how do you formulate a PDF in ROOT?

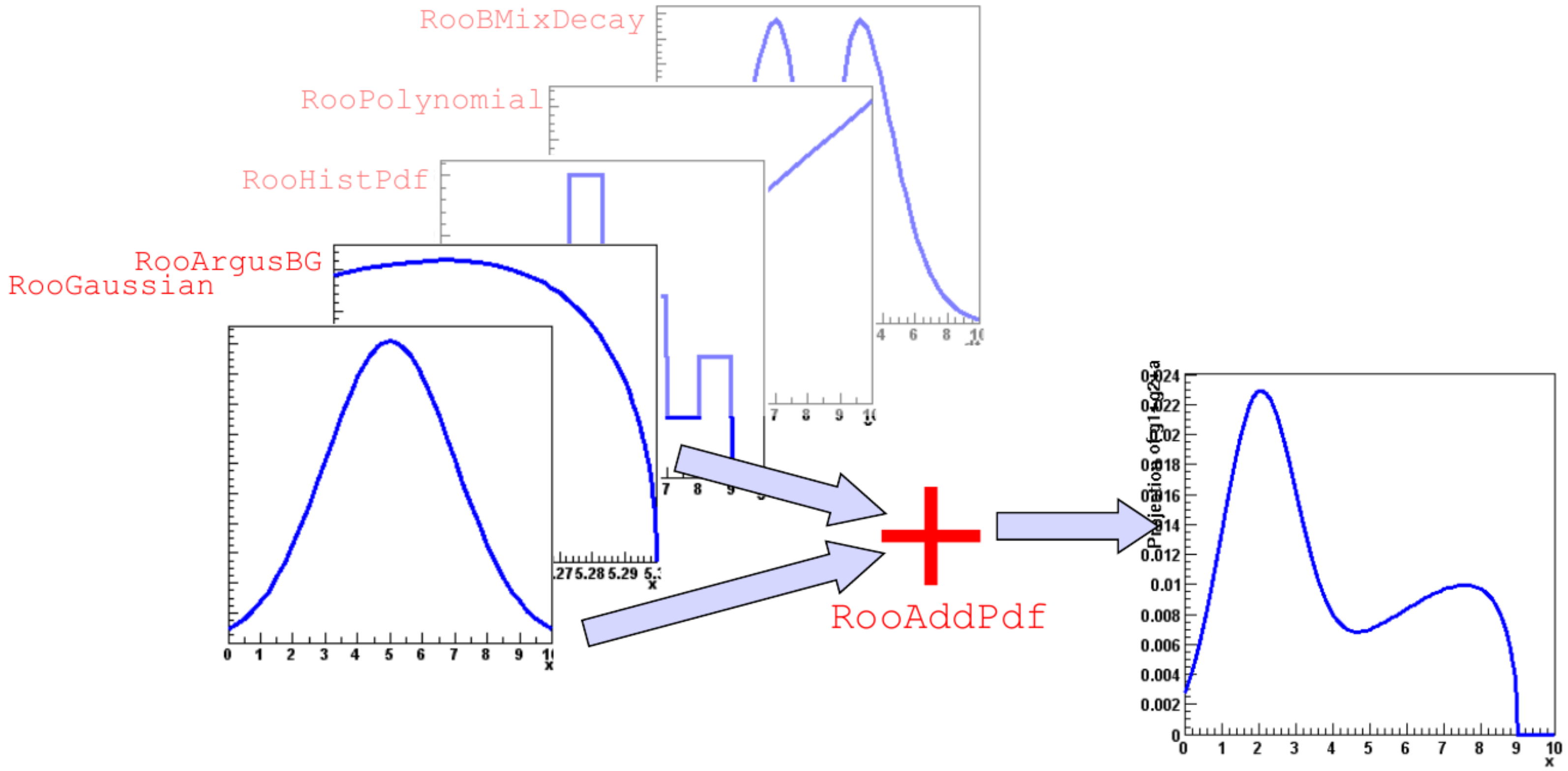


- Simple example: gauss (signal) + polynomial (bkg)
- Quickly becomes complicated: multidimensional, unbinned fits, non-trivial functions, non-analytic functions
- Core design philosophy:
  mathematical objects represented as C++ objects

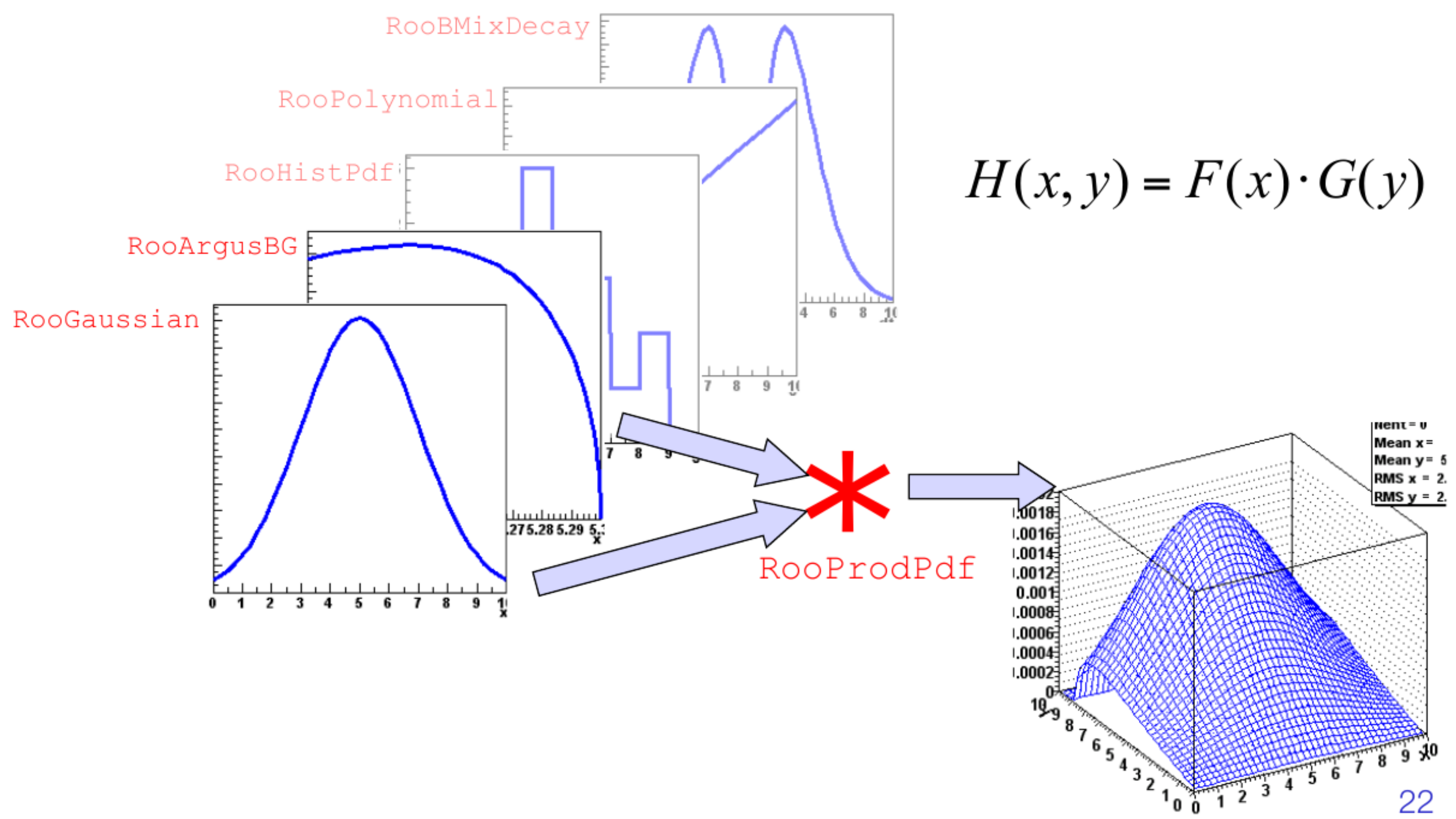
Mathematical concept		RooFit class
variable	x	`RooRealVar`
function	$f(x)$	`RooAbsReal`
PDF	$f(x)$	`RooAbsPdf`
space point	$\vec{x}$	`RooArgSet`
integral	$\int_{x_{\min}}^{x_{\max}} f(x)dx$	`RooRealIntegral`
list of space points		`RooAbsData`

RooFit - model building

- Easy to use standard components to build more complex/realistic models
- Addition



- Product (multi-dimensional)

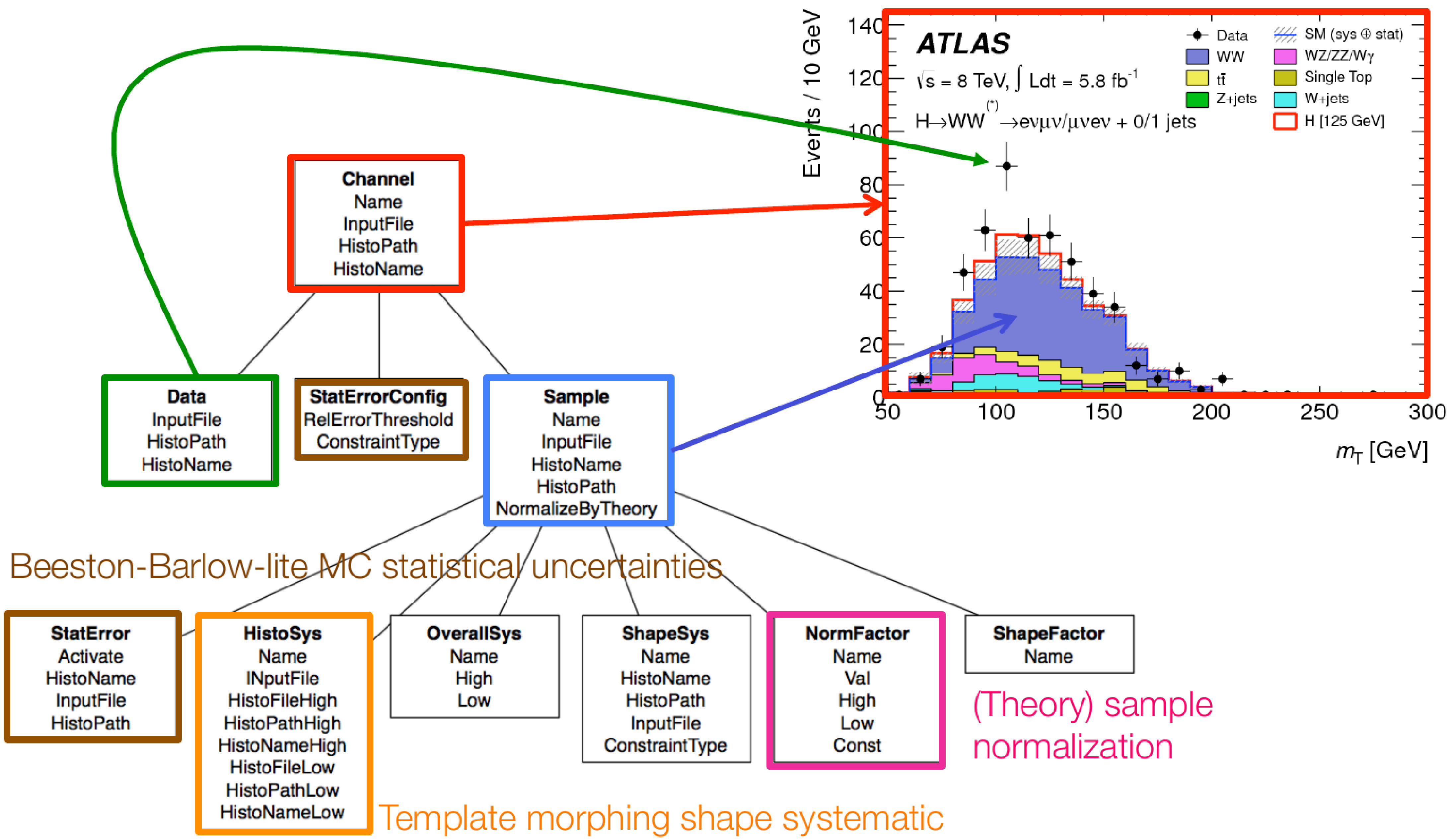


$$H(x, y) = F(x) \cdot G(y)$$

22

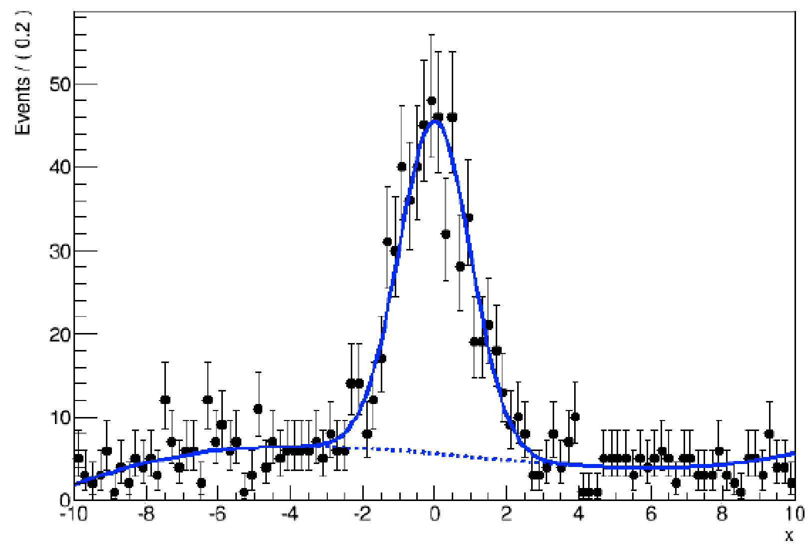
HistFactory

- Structured building of complex models based on binned templates (histograms)
- Classes of physics concepts:
  - Channel = region of phase space
    - One or more channels are combined to form a measurement
  - Sample = physics process: either data-driven or described by Monte Carlo (MC) simulation
  - Systematics = intrinsic uncertainty on your model



Systematics : nuisance parameters

- Empirical modeling of your model is easy to do, but expect some hard questions
  - Gaussian for signal + polynomial for background



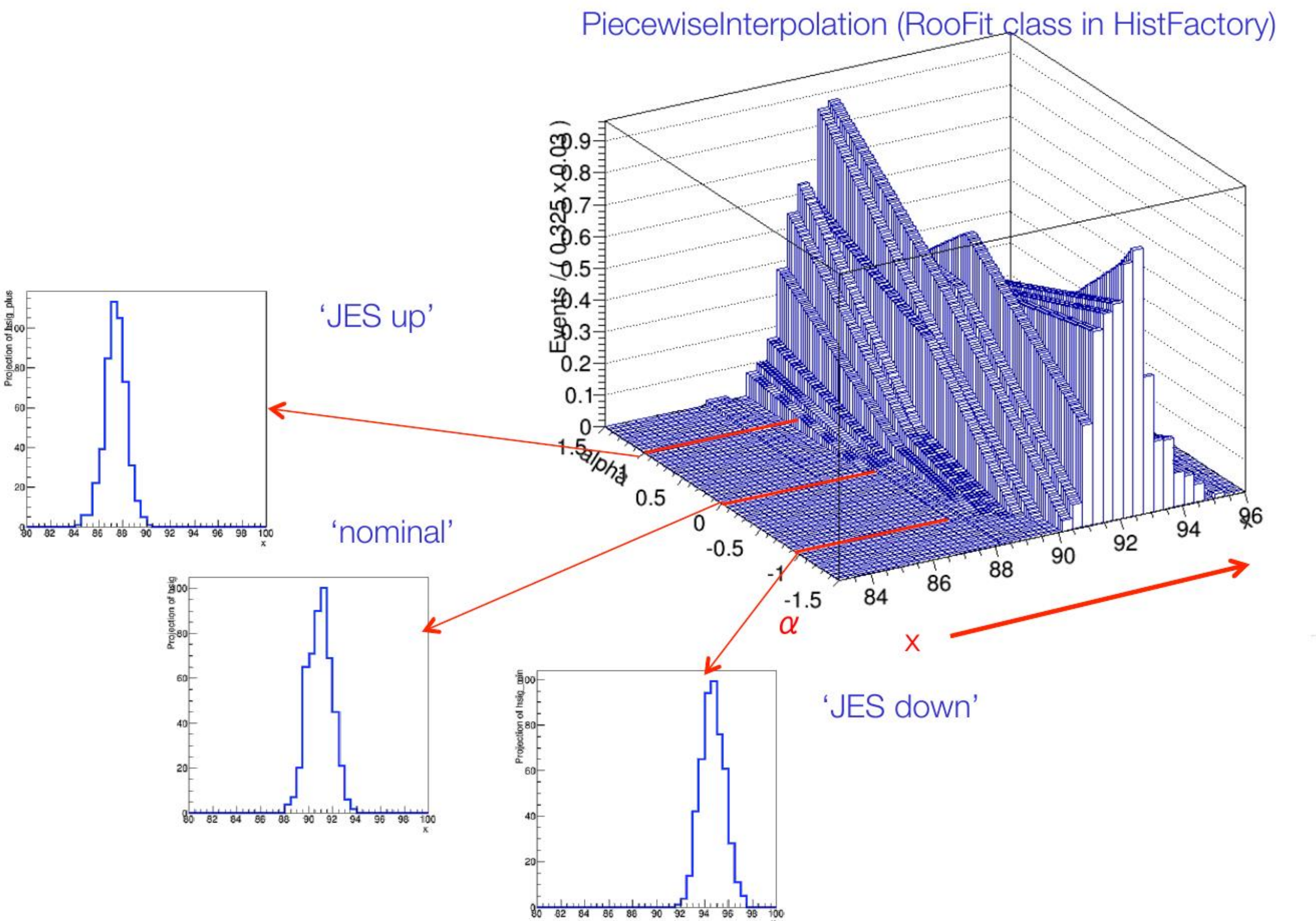
$$L(x \mid f, m, \sigma, a_0, a_1, a_2) = fG(x, m, \sigma) + (1 - f) Poly(x, a_0, a_1, a_2)$$

- Is your model correct?
  - Is the true signal distribution captured by a Gaussian?
- Is your model flexible enough?
  - Why use 4th order polynomial and not 6th order?
- How do your model parameters connect to known detector/theory uncertainties for your distribution?
  - What conceptual uncertainty does what parameter represent? And are all conceptual uncertainties represented?

Systematics modeling - interpolation

- A common solution is to introduce degrees of freedom in model that describe specific systematic/uncertainty!
- The +1/-1 σ variations sampled from MC simulation are compared to nominal MC response
    - (corrected/checked/double-checked to data by Perf. Groups)
- Interpolation, performed between **+1σ** ↔ **nominal** ↔ **-1σ** taken into the model as nuisance parameter

$$L(data \mid \mu, \theta) = Poisson(N_i \mid \mu \cdot s_i(\theta) + b_i(\theta)) \cdot p(\tilde{\theta}, \theta)$$



RooWorkspace

- Complete description of likelihood persistable in a ROOT file
- Factorizes building and using likelihood functions
  - In setup, team member, place and time

- Construct RooFit model `sum` and persist to ROOT file

```
RooWorkspace w("w") ;
w.import(sum) ;
w.writeToFile("model.root") ;
```

- Pass file to your colleague

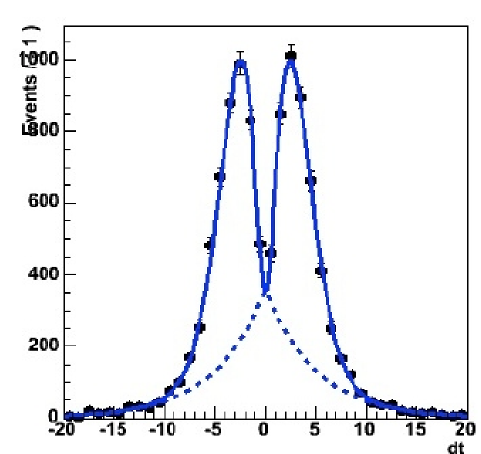
model.root

- Colleague resurrects likelihood, runs fit and produces plots

```
// Resurrect model and data
TFile f("model.root") ;
RooWorkspace* w = f.Get("w") ;
RooAbsPdf* model = w->pdf("sum") ;
RooAbsData* data = w->data("xxx") ;

// Use model and data
model->fitTo(*data) ;

RooPlot* frame =
         w->var("dt")->frame() ;
data->plotOn(frame) ;
model->plotOn(frame) ;
```



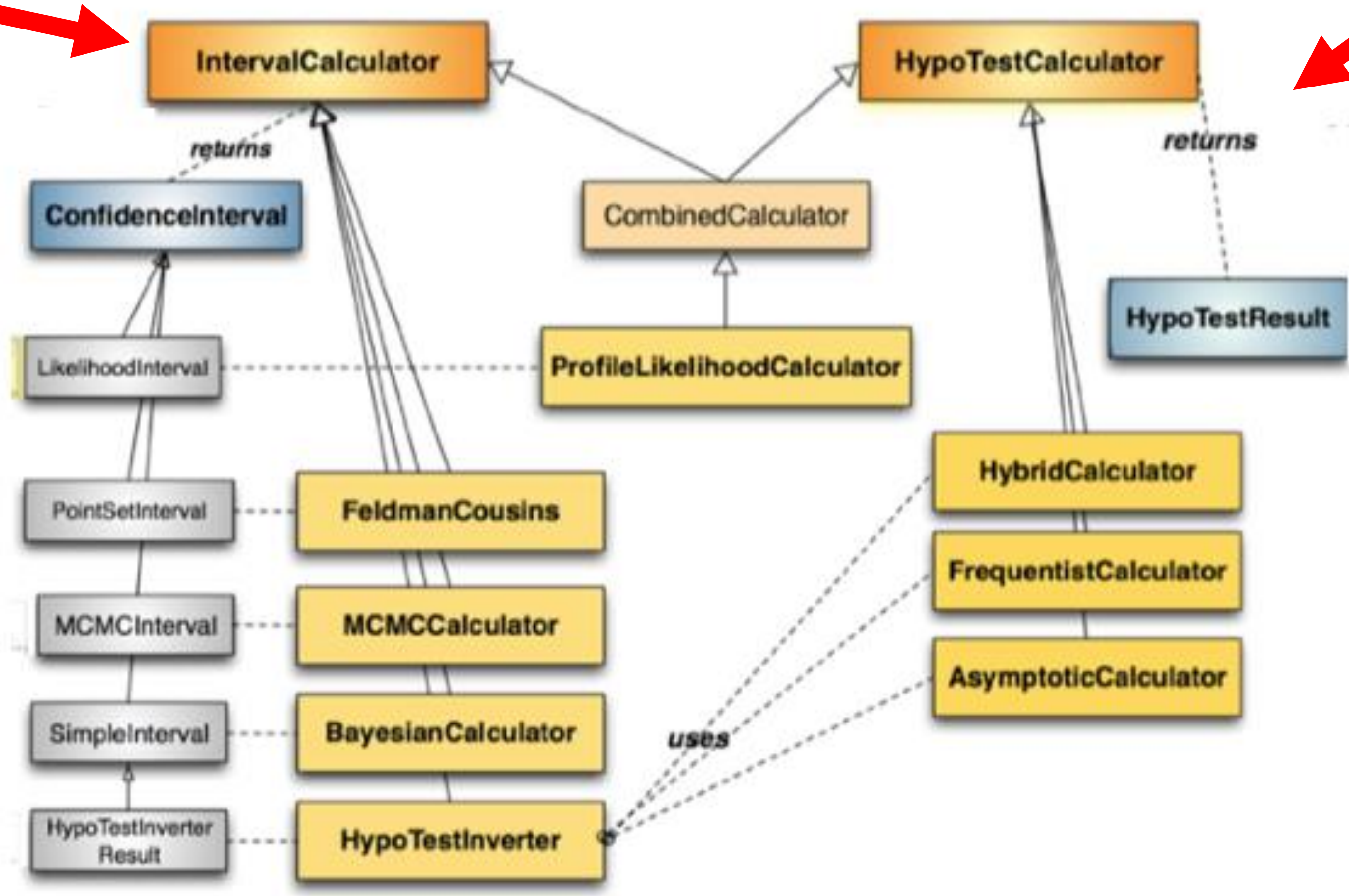
RooStats

- RooFit/HistFactory give tools to construct (complex) probability density functions
- RooWorkspace makes it possible to decouple statistical test tools from model contruction
- **RooStats** project/tools suite delivers a series of tools that can calculate intervals and perform hypothesis tests using a variety of statistical techniques
  - Frequentist/Bayesian/Likelihood-based methods (confidence/credible interval, hypothesis tests)

Confidence intervals: $[\theta_{-}, \theta_{+}]$, or θ <X at 95% C.L.
Hypothesis testing: → $p(data|\theta=0) = 1.10^{-7}$

RooStats class structure

- Abstract interface for procedure to calculate a confidence interval
- Abstract interface for result: *s>6.8 is at 95% C.L.*



- Abstract interface for hypothesis tester to calculate a p-value
- Concrete result class: *HypoTestResult*
  $p_b = 0.023$
  $CL_S = 0.00513$

Overview

- **Step-0:** define signal/control/validation regions
  - Input TTrees (derived from xAOD), histograms, numbers
- **Step-1:** Construct PDF and the likelihood function

  RooFit or HistFactory + RooFit
  - Result from data is a distribution
  - Model signal and background by PDF (prob. density func.)
  - Construct likelihood(s) by joining data and model(s)
- ↓
- **RooWorkspace**
- ↓
- **Step-2:** Statistical tests on parameter of interest **μ**

  RooStats
  - Construct test statistic q_μ from likelihoods
  - Obtain expected distributions of q_μ for various **μ** values
  - Determine discovery $\mathbf{p_0}$ and signal exclusion limit
- **Step-3:** Repeat for each model (assumed value m_H)

HistFitter

- adds steps-0 and 3
- allows full analysis chain from simple configuration file

Links

- RooFit overview (2004): http://www.nikhef.nl/~verkerke/talks/chep03/chep2003_v4.pdf
- ATLAS Statistics Forum page on Stat. Tools: https://twiki.cern.ch/twiki/bin/viewauth/AtlasProtected/StatisticsTools
- RooFit/RooStats at ACAT 2014: https://indico.cern.ch/event/258092/session/0/contribution/140/material/slides/1.pdf
- Higgs Combination procedure/explanation of CLs observed/expected and error bands: http://cds.cern.ch/record/1375842
- HistFactory documentation:
  https://cdsweb.cern.ch/record/1456844/
  https://twiki.cern.ch/twiki/bin/view/RooStats/HistFactory

[23] K. Cranmer, G. Lewis, L. Moneta, A. Shibata, W. Verkerke, HistFactory: A tool for creating statistical models for use with RooFit and RooStats, CERN-OPEN-2012-016.

[24] L. Moneta, K. Belasco, K. S. Cranmer, S. Kreiss, A. Lazzaro, et al., The RooStats Project, PoS ACAT2010 (2010) 057. arXiv:1009.1003.

[25] W. Verkerke, D. P. Kirkby, The RooFit toolkit for data modeling, eConf C0303241 (2003) MOLT007. arXiv:physics/0306116.

[26] R. Brun, F. Rademakers, ROOT: An object oriented data analysis framework, Nucl.Instrum.Meth. A389 (1997) 81–86. doi:10.1016/S0168-9002(97)00048-X.

[27] I. Antcheva, M. Ballintijn, B. Bellenot, M. Biskup, R. Brun, et al., ROOT: A C++ framework for petabyte data storage, statistical analysis and visualization, Comput.Phys.Commun. 182 (2011) 1384–1385. doi:10.1016/j.cpc.2011.02.008.

HistFitter introduction

Introduction

- **HistFitter** is a statistical tool/framework used in (almost) all SUSY WG analyses since 2012 for fitting, interpretation and presentation of fit results
  - Developed in SUSY strong production 1-lepton group, quickly adopted as recommended tool
  - Small core team: Max Baak, Geert-Jan Besjes, David Cote, Alex Koutsman, Jeanette Lorenz and Dan Short
  - Also used (more and more) in Higgs, Exotics and Top WGs

- **HistFitter** is:
  - built on top of RooFit/HistFactory and RooStats
  - consists of Python part for configuration and C++ part for CPU-intensive calculations

- Why HistFitter?
- **HistFitter** extends RooFit/HistFactory and RooStats in four key areas:
  - <u>Programmable framework:</u> performing complete analysis (steps 0-4) from a simple configuration file
  - <u>Analysis strategy:</u> common physics analysis strategy concepts, such as control/signal/validation regions, woven into the fabric of HistFitter design
  - <u>Bookkeeping:</u> can keep track of numerous data models, from histogram production until final statistical tests → handy when working with large collections of signal hypotheses (*signal grids*)
  - <u>Presentation and interpretation:</u> multiple methods are provided to determine statistical significance of signal hypotheses, and produce publication-quality tables and plot summarizing the fit results (*step 4*)

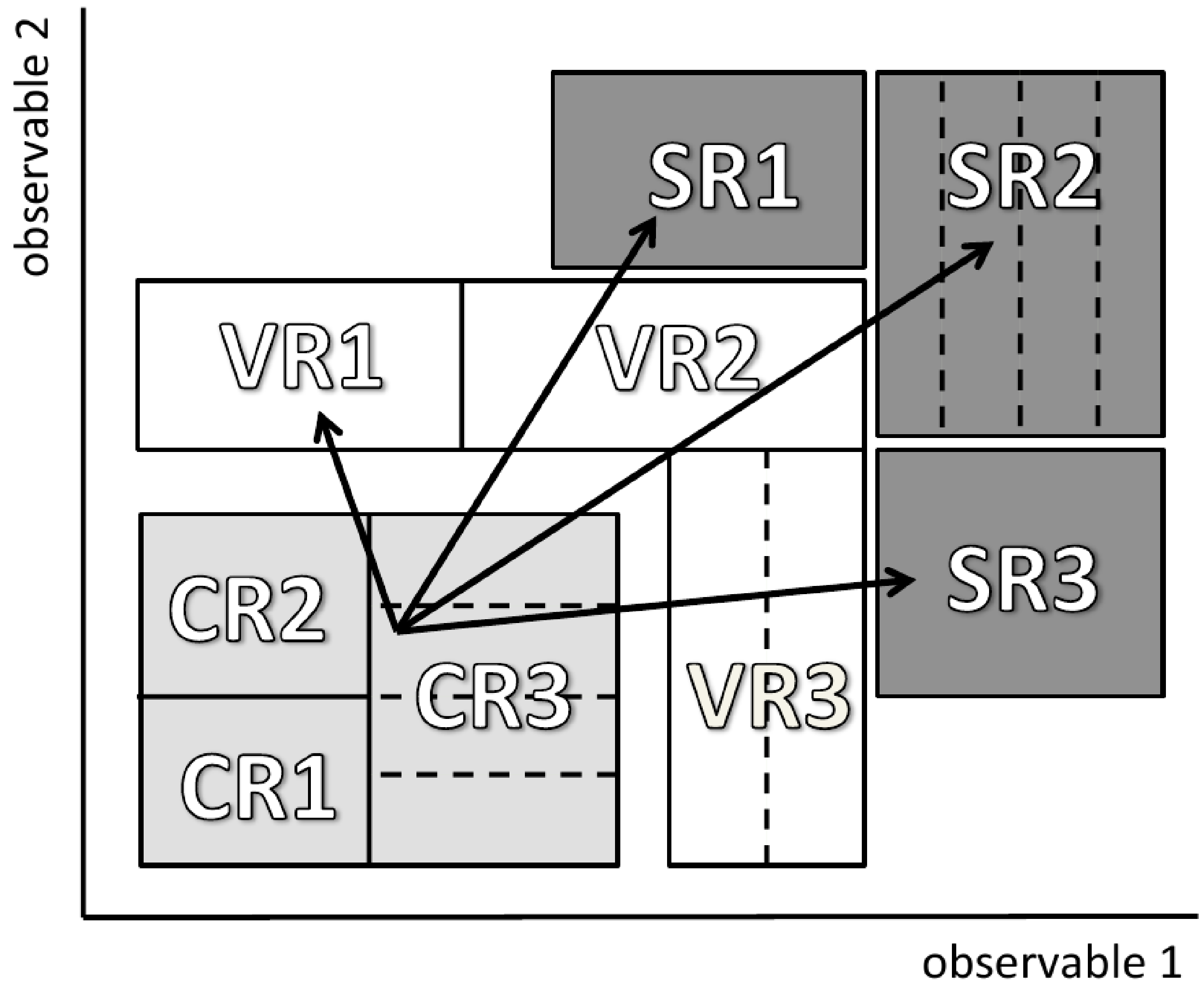
Data analysis strategy

- Particle physics analyze large data samples for measurements of discovery
- Data interpretation relies on using external - simulation, Monte Carlo (MC) - predictions for backgrounds and signal
- HistFitter configures and builds parametric models from these predictions

- Typically one defines several phase space regions to study a specific phenomenon

- Definition depends on the purpose:
  - **Signal region:** signal-rich region (SR)
  - **Control region:** background-rich region (CR), fit simulated backgrounds to data
  - **Validation region:** validation of extrapolation (VR)

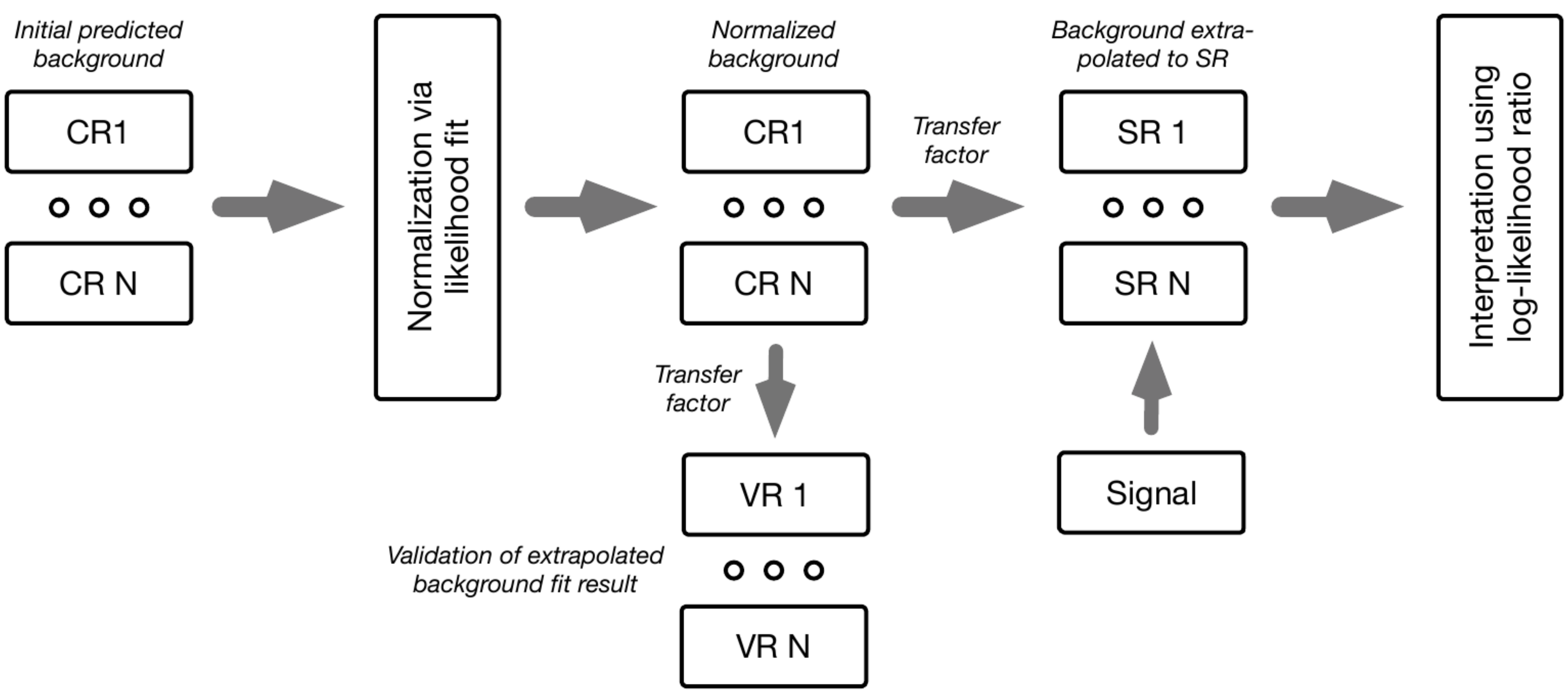
- Concepts of CR/SR/VR woven into the fabric of HistFitter



Analysis strategy flow

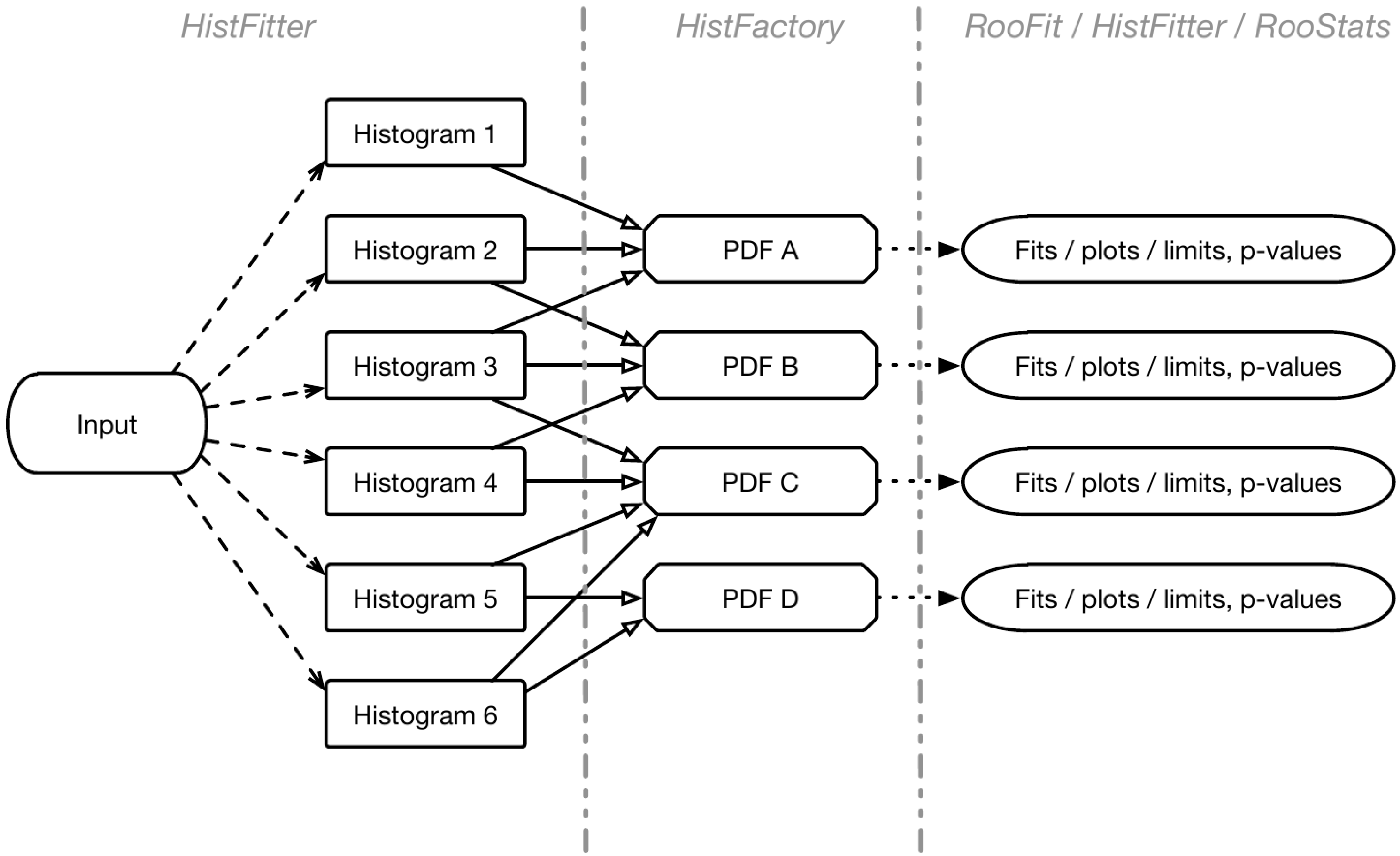
- Each CR/VR/SR modeled by a separate PDF, combined in a simultaneous fit
- Parameters shared in all regions → consistent background/signal prediction and systematics
  - Sharing user-defined

- Analysis flow:
  - Backgrounds normalized to data in a fit of control regions
  - Extrapolate to validation/signal regions using transfer factors (ratio of events between CR and SR/VR)
  - If good agreement in VR, unblind the SR
  - If no excess, add signal prediction and interpret/set limits



Processing sequence

- Based on user-defined configuration file, processing sequence of HistFitter split in three stages



step-0	**step-1**	**step-2/3**
Histogram production	PDF construction Workspace building	Analysis of models

Model construction

- Models constructed using HistFactory from input histograms
- General form of the constructed likelihood:

$$L(\boldsymbol{n}, \boldsymbol{\theta}^0 | \mu_{\mathrm{sig}}, \boldsymbol{b}, \boldsymbol{\theta}) \;=\; P_{\mathrm{SR}} \;\times\; P_{\mathrm{CR}} \;\times\; C_{\mathrm{syst}}$$

  - P = Poisson measurements of number of observed events in CR/SR (VR)
  - C = Constraint terms for systematic uncertainties, auxiliary measurements
  - Likelihood depends on number of observed events in all regions (n), predictions for various background processes (b), the nuisance parameter (θ) parametrizing the systematic uncertainties with their central value (θ^0) and signal strength (μ_{SIG})

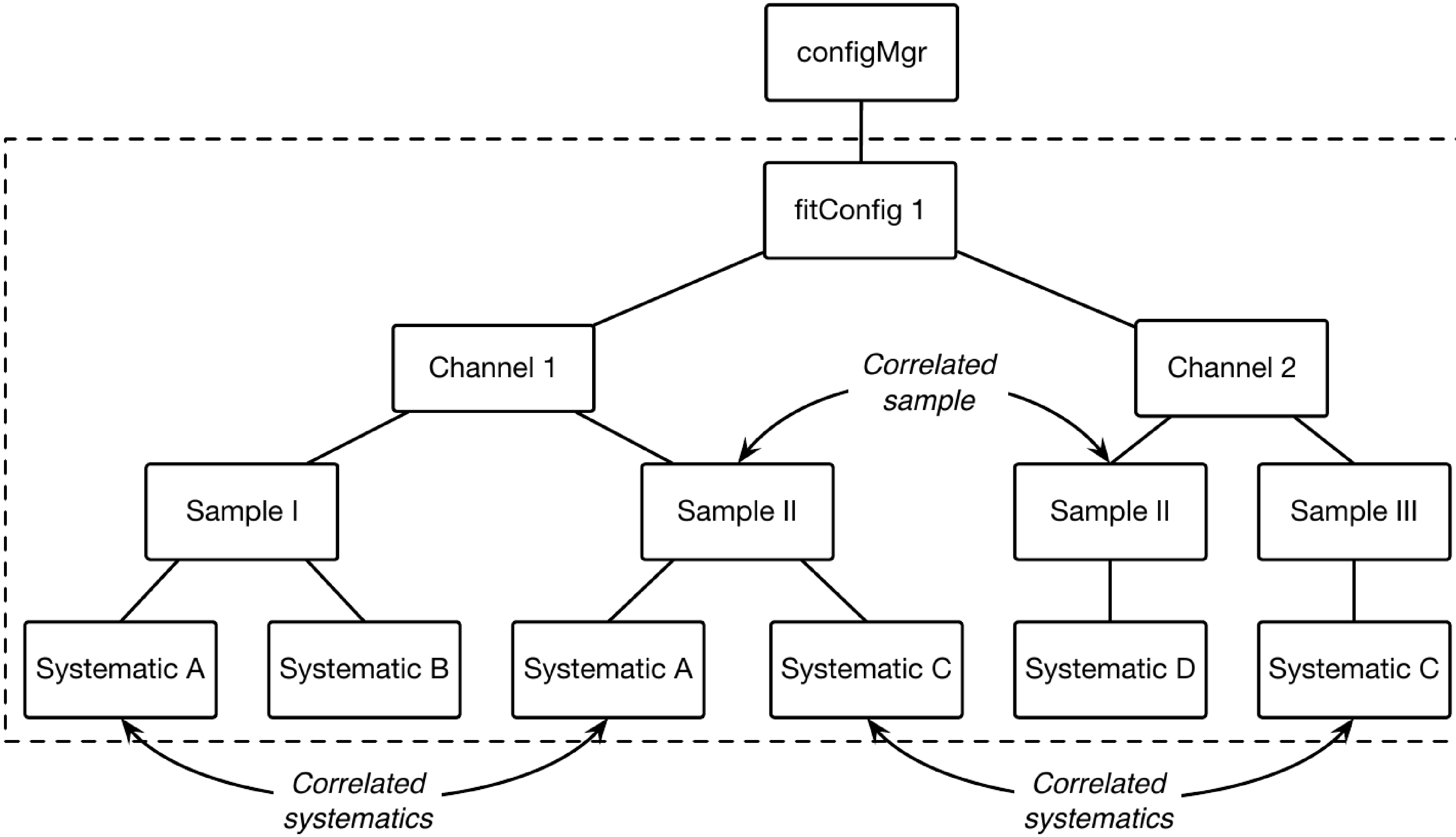
- Likelihood has multiple building blocks:
  - Control/validation/signal regions: called `channel` in HistFitter (HistFactory)
  - Signal and background processes: called `sample` in HistFitter (HistFactory)
  - Uncertainties: called `systematic` in HistFitter (HistFactory)
    - Including statistical/theory/experimental uncertainties

- HistFitter is designed to build and manipulate PDFs of nearly arbitrary complexity
- Bookkeeping/configuration machinery realized through a user-defined Python configuration file

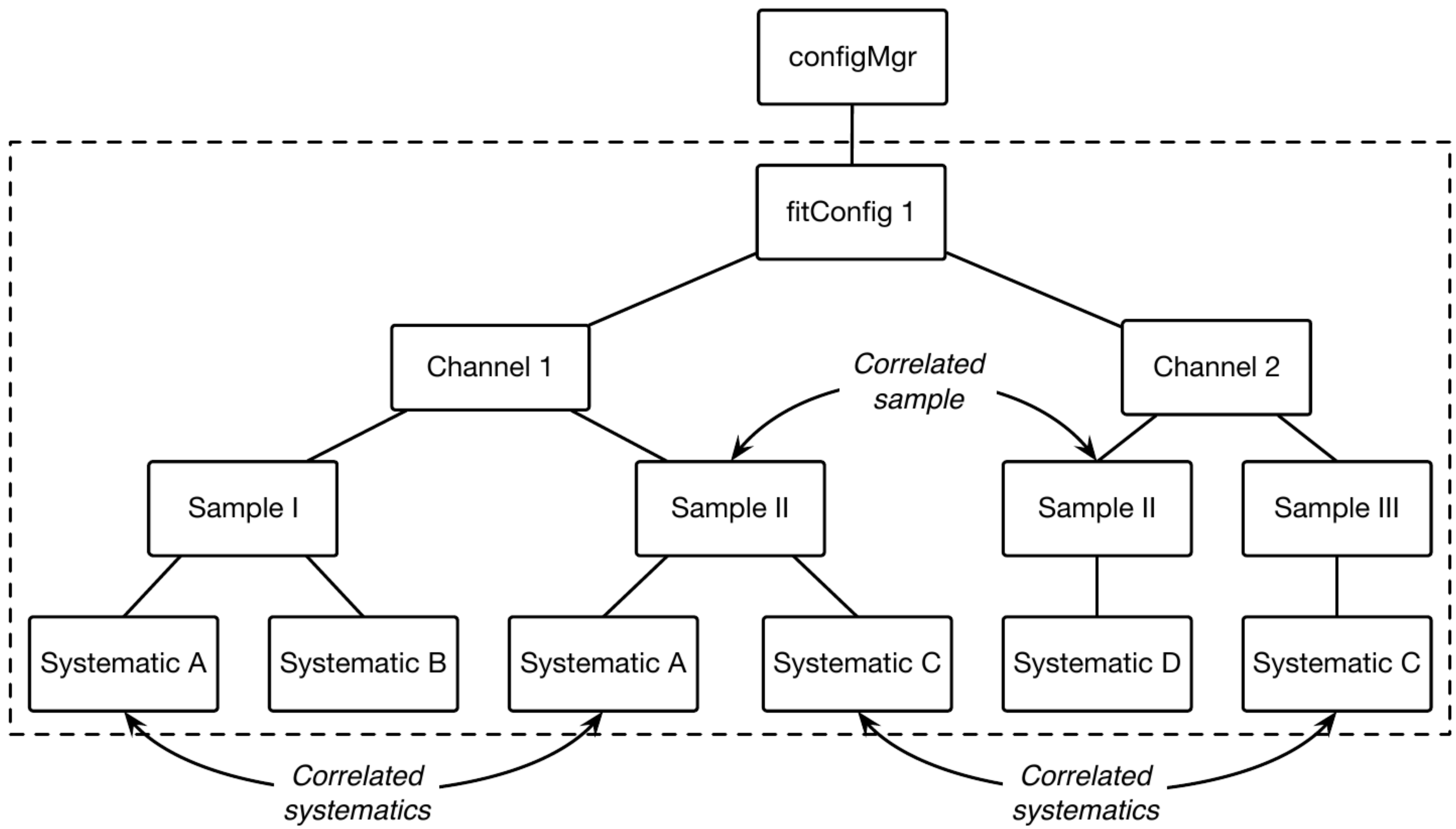
- Configuration manager (`configManager`) highest level (singleton) object in Python and C++
- Manages `fitConfig` objects that contain PDF and meta-data

Fit configuration

- `fitConfig` objects summarize `channels`, `samples` and `systematics` together with corresponding input histograms



Fit configuration properties



- `fitConfig`: can be cloned/extended (see next slide)
- `channels:` either single-bin or multi-bin (shape), property as CR/VR/SR
- `samples:` input from TTree, TH1 or raw (hard-coded) floats, correlated between channels
- `systematics`: provided as ±1σ variation of nominal histogram; input from TTree, TH1 or raw floats; can be correlated between samples and/or channels; many types available extended from HistFactory base types (see later); trickle-down mechanism (see backup)

Common fit strategies

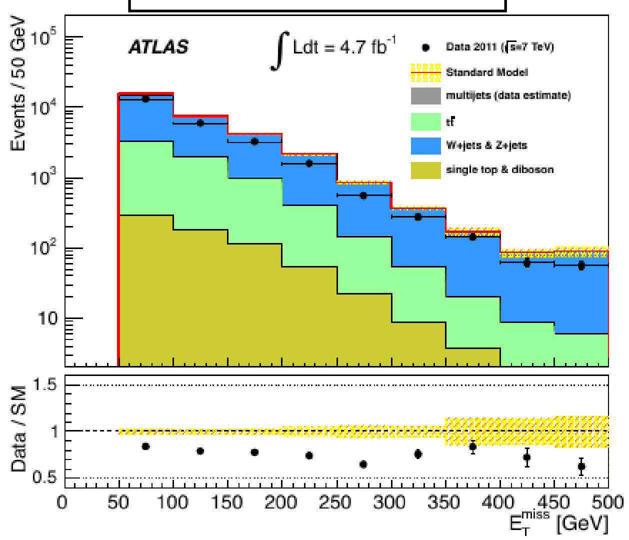
- **Background-only fit**: estimate background yields in validation/signal regions; including *only* CRs in the fit to data; no signal component included in fit configuration
- **Model-dependent signal fit**: set exclusion limit on a specific signal model; possible use of multi-binned (or multi-SR) shape fit for a robust signal estimation - aka ***exclusion fit***
- **Model-*in*dependent signal fit**: to obtain model-independent upper limits on number of BSM events beyond background prediction; only usable with one single-bin SR (otherwise not model-independent) - aka ***discovery fit***

Fit setup	*Background-only fit*	*Model-dependent signal fit*	*Model-independent signal fit*
Samples used	backgrounds	backgrounds + signal	backgrounds + dummy signal
Fit regions	CR(s)	CR(s) + SR(s)	CR(s) + SR

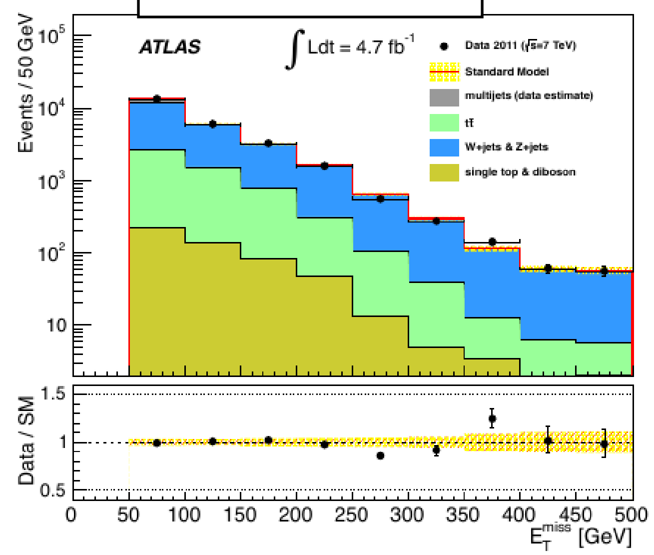
Presentation of results

- HistFitter includes a collection of tools (scripts/functions) to present/understand fit results

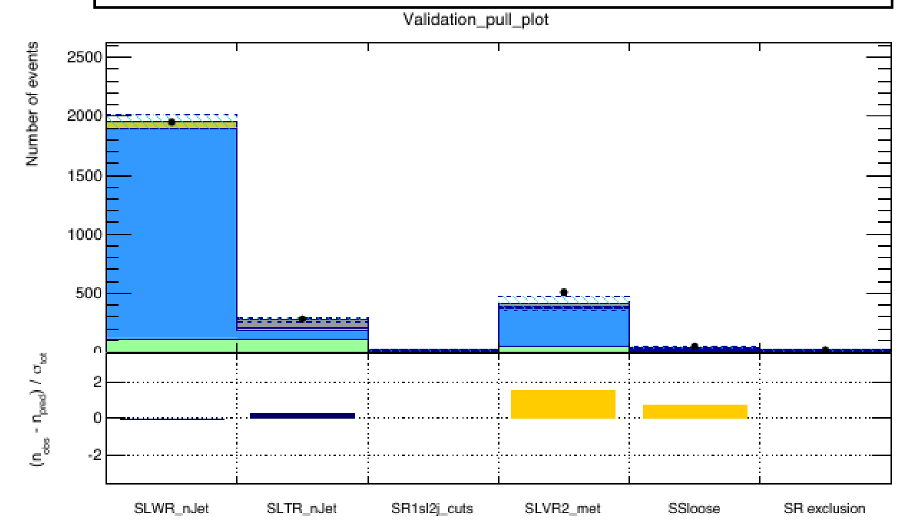
Before Fit



After Fit



After Fit VRs pull plot



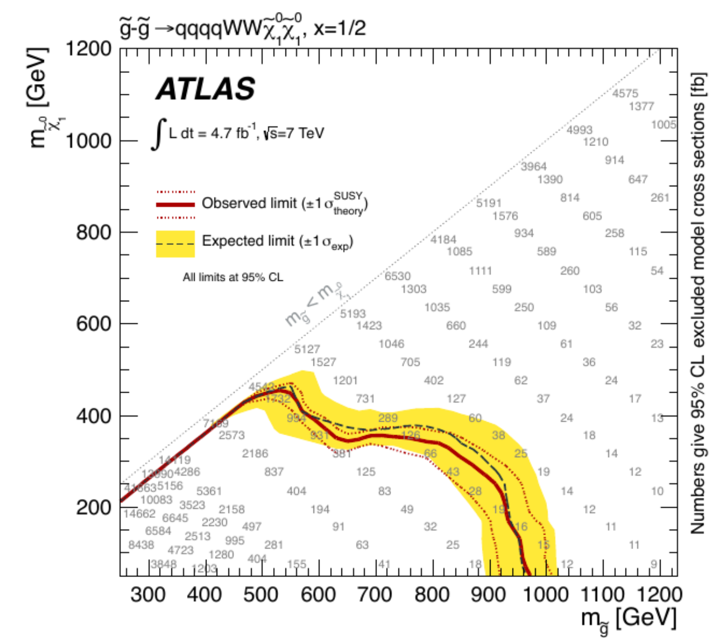
Yields Table

Signal Region	SR1	SR2
Observed events	16	19
Fitted bkg events	19.54 ± 3.93	20.47 ± 5.14
Fitted Top events	4.02 ± 0.96	4.32 ± 1.04
Fitted V+jets events	9.89 ± 1.86	10.47 ± 1.91
Fitted other background events	1.14 ± 0.15	1.19 ± 0.16
Fitted QCD events	4.49 ± 2.72	4.49 ± 4.24
MC exp. SM events	24.85	26.32
MC exp. Top events	8.42	9.11
MC exp. V+jets events	10.82	11.55
MC exp. other background events	1.13	1.17
Data-driven exp. QCD events	4.49	4.49

Systematics Table

Uncertainty of channel	SR1	SR2
Total background expectation	19.54	20.47
Total statistical ($\sqrt{N_{\rm exp}}$)	± 4.42	± 4.52
Total background systematic	± 3.93 [20.14%]	± 5.14 [25.09%]
QCD background	± 2.66	± 4.20
Statistical uncertainties	± 2.54	± 1.86
Jet Energy Scale	± 1.15	± 1.17
Top yield	± 0.82	± 0.88
Renormalization scale (Top)	± 0.34	± 0.39
V+jets yields	± 0.28	± 0.29
Renormalization scale (V+jets)	± 0.14	± 0.03

Exclusion contour with upper limits



Model-independent upper limits

Signal channel	$\langle\sigma_{\rm vis}\rangle^{95}_{\rm obs}$[fb]	$S^{95}_{\rm obs}$	$S^{95}_{\rm exp}$	$p(s=0)$
SR3b	0.19	3.9	$4.4^{+1.7}_{-0.6}$	0.50
SR0b	0.80	16.3	$8.9^{+3.6}_{-2.0}$	0.03

HistFitter & documentation

- HistFitter paper on arXiv: http://arxiv.org/abs/1410.1280
- HistFitter webpage with doxgen documentation: http://cern.ch/histfitter
- Tutorial (to be discussed next): https://twiki.cern.ch/twiki/bin/view/Main/HistFitterTutorialOutsideAtlas
- ACAT 2014 talk on HistFitter: https://indico.cern.ch/event/258092/session/8/contribution/39

HistFitter tutorial

Running HistFitter

- `HistFitter.py <options> <configuration_file>`

- **-t:** Create histograms in all regions used for all backgrounds, signal, data from TTrees
- **-w**: Build workspaces from histograms
- **-f:** Fit
- **-D**: various drawing options, to be discussed later
- **-L:** log level {VERBOSE,DEBUG,INFO,WARNING,ERROR,FATAL,ALWAYS}
- **-m PARAM**: run Minos for asymmetric error calculation
  - optionally give parameter names comma separated; for all parameters use ‘ALL’ or ‘all’
- **-l**: Calculate upper limit
- **-p**: Calculate the CLs value for a specific signal model (for exclusion)
- **-i**: interactive mode, keeps you in python command line, but shows plots on your screen

- To see all options run: `HistFitter.py --help`

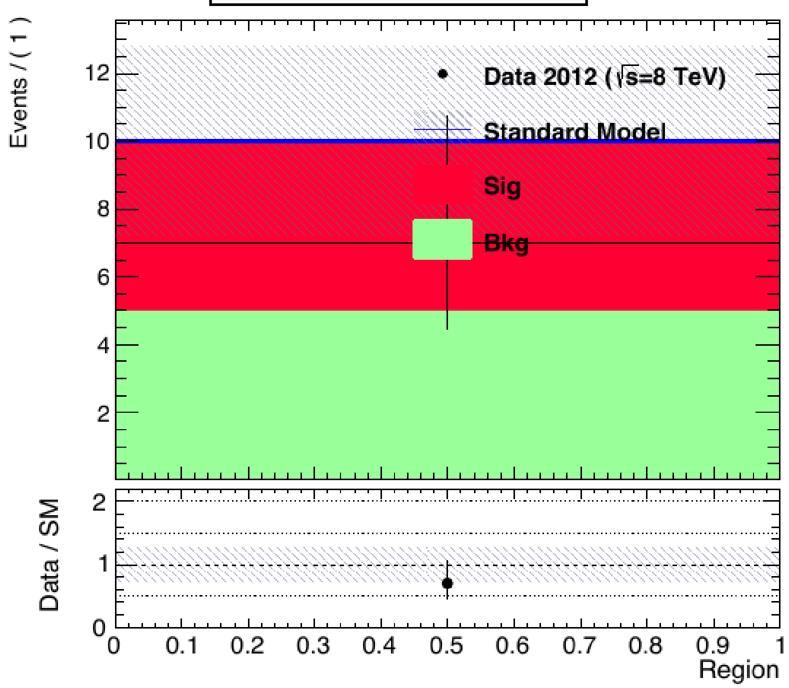
Simple example

- Simple example with one region with one bin:

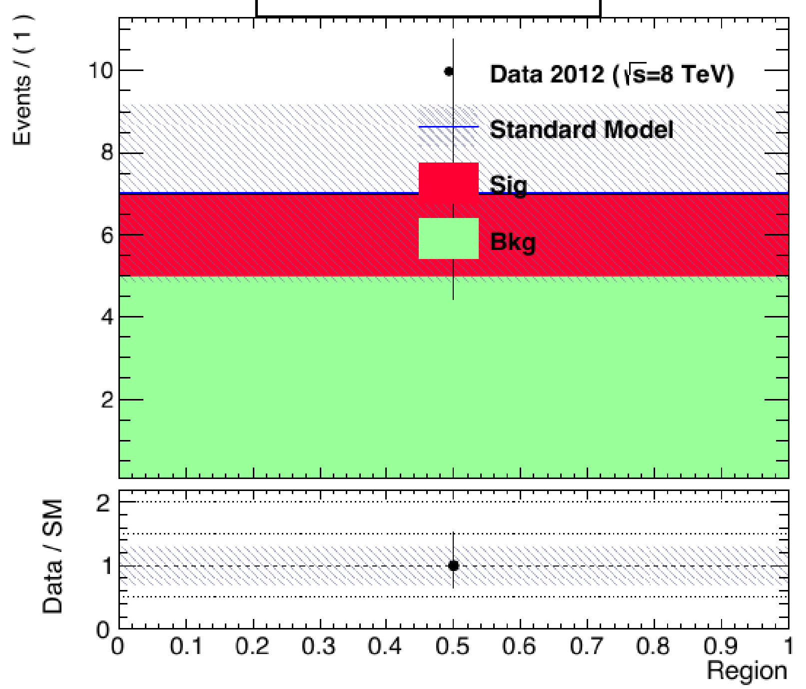
```
HistFitter.py -w -f -D "before,after,corrMatrix" -i
analysis/tutorial/MyUserAnalysis.py
```

- Creates the workspace
- Runs the fit
- Plots before/after fit regions and correlation matrix
- Keeps you in interactive mode

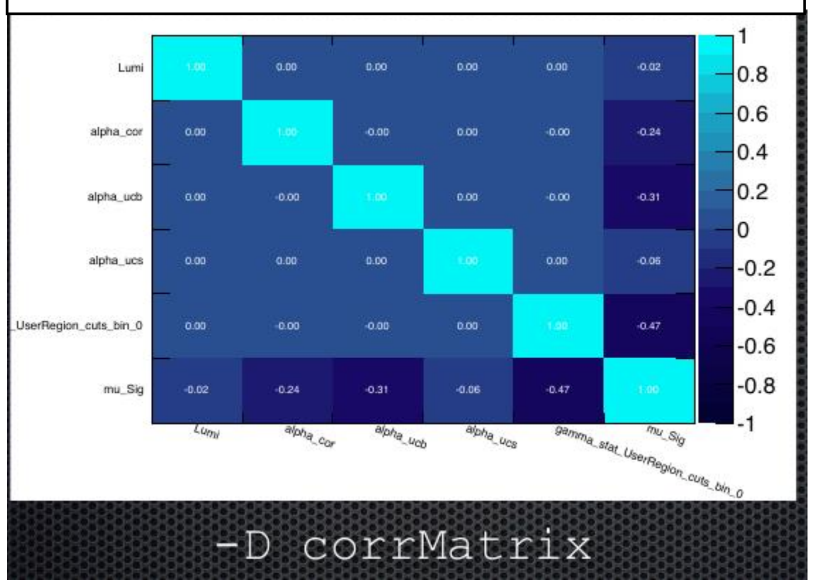
Before Fit



After Fit



correlations matrix



Config file explained - I

- Define a configManager and setup a `fitConfig` ana named SPlusB
- ```
  from configManager import configMgr
  ana = configMgr.addFitConfig("SPlusB")
  ```

- Add one channel/region to the `fitConfig`
- `chan = ana.addChannel("cuts",["UserRegion"],1,0.5,1.5)`

- One defines the region/channel in cutsDict (as one would in ROOT for TTree call)
- Here include all:
- `configMgr.cutsDict["UserRegion"] = "1."`

- Channels can also be binned (shape-fit)
- ```
  chan = ana.addChannel("myObs", ["mySelection"], nBins, varLow,
  varHigh)
  ```

Config file explained - II

- Define samples: bkgSample, sigSample and dataSample

- `#` Define samples
  `bkgSample = Sample("Bkg",kGreen-9)` # define a background sample with color KGreen-9 if plotting
  `bkgSample.setStatConfig(True)` #This sample gets statistical uncertainties
  `bkgSample.buildHisto([nbkg],"UserRegion","cuts")` #Build histograms from numbers defined by the user
  `bkgSample.buildStatErrors([nbkgErr],"UserRegion","cuts")`

  `sigSample = Sample("Sig",kPink)` #A signal sample with color kPink
  `sigSample.setNormFactor("mu_Sig",1.,0.,100.)` # This samples receives a normalization parameter
  `sigSample.setStatConfig(True)` #This sample gets statistical uncertainties
  `sigSample.setNormByTheory()` # and uncertainties due to the luminosity are added
  `sigSample.buildHisto([nsig],"UserRegion","cuts")`
  `sigSample.buildStatErrors([nsigErr],"UserRegion","cuts")`

  `dataSample = Sample("Data",kBlack)` #Data sample
  `dataSample.setData()`
  `dataSample.buildHisto([ndata],"UserRegion","cuts")`

  # add all samples to the fitconfig object and thus to all channels
  `ana.addSamples([bkgSample,sigSample,dataSample])`

Config file explained - III

- Add systematics to signal/background samples
- Correlating systematics happens by giving them the same name

- # Set uncorrelated systematics for bkg and signal (1 +- relative uncertainties)

```
ucb = Systematic("ucb", configMgr.weights, 1.2,0.8, "user","userOverallSys")
ucs = Systematic("ucs", configMgr.weights, 1.1,0.9, "user","userOverallSys")
# correlated systematic between background and signal (1 +- relative uncertainties)
corb = Systematic("cor",configMgr.weights, [1.1],[0.9], "user","userHistoSys")
cors = Systematic("cor",configMgr.weights, [1.15],[0.85],
"user","userHistoSys")

bkgSample.addSystematic(corb)
bkgSample.addSystematic(ucb)
sigSample.addSystematic(cors)
sigSample.addSystematic(ucs)
```

Table production

- **YieldsTable.py** produces customizable tables of yields before/after fit
- Example: `YieldsTable.py -s Top,WZ,BG,QCD -c SLWR_nJet,SLTR_nJet -w results/MyConfigExample/BkgOnly_combined_NormalMeasurement_model_afterFit.root -o MyYieldsTable.tex`

table.results.yields channel	SLWR_nJet	SLTR_nJet	SR1sl2j	SS_metmeff2Jet
Observed events	1794	269	25	26
Fitted bkg events	1800.73 ± 39.91	262.45 ± 11.47	28.53 ± 5.26	31.74 ± 8.50
Fitted Top events	117.20 ± 11.42	113.20 ± 12.53	6.17 ± 1.12	6.65 ± 1.26
Fitted WZ events	1629.37 ± 42.19	69.75 ± 6.63	13.95 ± 2.03	14.57 ± 1.98
Fitted BG events	43.49 ± 1.90	23.19 ± 1.94	0.96 ± 0.32	1.00 ± 0.32
Fitted QCD events	10.64 ± 0.51	56.30 ± 13.65	7.44 ± 3.75	9.52 ± 7.54
MC exp. SM events	1921.26	261.96	32.04	35.35
MC exp. Top events	165.16	153.98	8.75	9.38
MC exp. WZ events	1647.04	66.30	15.26	15.82
MC exp. BG events	40.96	25.03	0.59	0.63
data-driven exp. QCD events	68.06	16.64	7.44	9.52

- **SysTable.py** produces customizable tables of systematic breakdown per region (or sample)
- Example: `SysTable.py -w results/MyConfigExample/BkgOnly_combined_NormalMeasurement _model_afterFit.root -c SR1sl2j -o systable_SR1sl2j.tex`

Uncertainty of channel	SR1sl2j
Total background expectation	28.53
Total statistical ($\sqrt{N_{\mathrm{exp}}}$)	± 5.34
Total background systematic	± 5.26 [18.43%]
gamma_stat_SR1sl2j_cuts_bin_0	± 3.63
alpha_QCDNorm_SR1sl2j	± 3.63
alpha_JES	± 0.93
mu_Top	± 0.65
alpha_KtScaleTop	± 0.52
alpha_KtScaleWZ	± 0.37
mu_WZ	± 0.36

Signal model hypothesis test

- Once you have unblinded your SR, one can calculate the CLs/p-value on specific signal models using the exclusion fit (aka model-dependent fit setup)
- As simple in HistFitter as calling:

```
HistFitter.py -p analysis/tutorial/MyUserAnalysis.py
```

- Will calculate:
  - CLs_observed = taking N observed events as data in all regions
  - CLs_expected = taking N expected events as data in all regions
  - CLs_expected ±1sigma experimental uncertainty = N expected as data, ±1sigma fit results
    - yellow band next slide
  - CLs_observed ±1sigma signal theory uncertainty = N observed as data, ±1sigma signal theory
    - need to set the name of the signal theory uncertainty systematic as `Systematic("SigXSec", ...)`
    - red-dotted lines next slide

- Setting calculator and test statistic type can be set in configManager (see backup):

```
## setting the parameters of the hypothesis test
#configMgr.nTOYs=5000
configMgr.calculatorType=2 # 2=asymptotic calculator, 0=frequentist calculator
configMgr.testStatType=3   # 3=one-sided profile likelihood test statistic (LHC default)
configMgr.nPoints=20       # number of values scanned of signal-strength for upper-limit
determination of signal strength.
```

- Result of ‘-p’ stored in a ROOT file with ‘hypotest’ in the name:

```
results/MySimpleChannelAnalysis_fixSigXSecNominal_hypotest.root
```

Contour plot explained

- https://twiki.cern.ch/twiki/bin/view/AtlasProtected/SUSYLimitPlotting

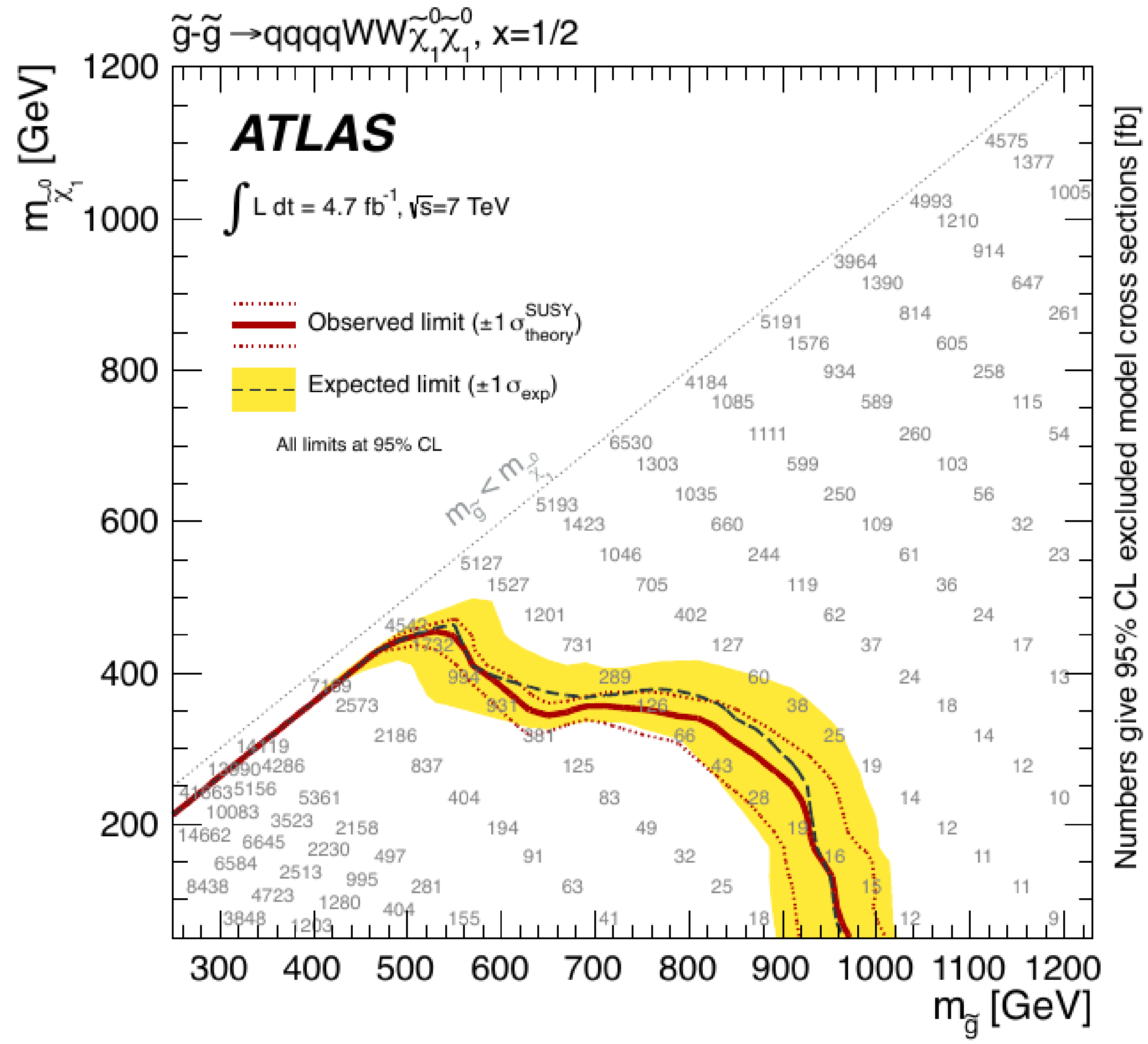
Description of limit lines

The model limits should be computed using the HistFitter package. We present the following limits:

1. **Observed limit** (thick solid dark-red line): all uncertainties are included in the fit as nuisance parameters, with the exception of the theoretical signal uncertainties (PDF, scales).
2. **Expected limit** (less thick long-dashed dark-blue line): all uncertainties are included in the fit as nuisance parameters, with the exception of the theoretical signal uncertainties (PDF, scales).

We present the following uncertainty bands:

- **±1σ lines around observed limit** (1) with style "thin dark-red dotted": re-run limit calculation (1) while increasing or decreasing the signal cross section by the theoretical signal uncertainties (PDF, scales).
- **±1σ band around expected limit** (2) with style "yellow band": the band contours are the ±1σ results of the fit (2).

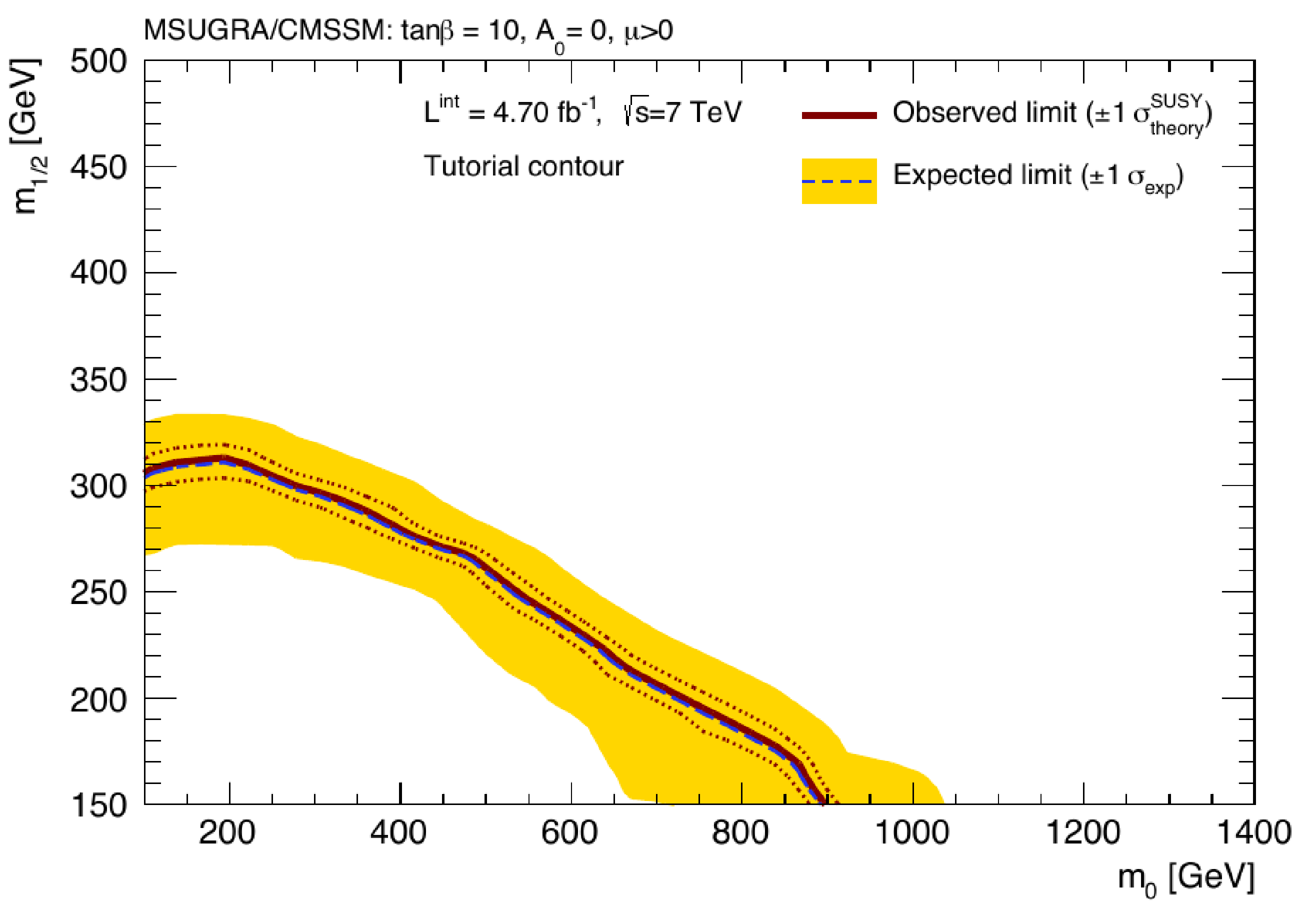


Contour plot production

- Typically a grid of signal model points with varying signal parameters (m_H or m_{gluino}) get processed to produce an exclusion contour
- Five steps to produce (Part 5 of tutorial):

1. run hypothesis tests over all grid points (results saved in multiple *hypotest* files)
2. merge all the output root files into one using `hadd` (if stored in a separate files)
3. transform this set of hypothesis tests into a plain-text file: `makelistfiles.C`
4. create TH2D(s) from the ascii data in this list file: `makecontourhists.C`
5. plot TH2D(s) to draw contour lines and cosmetics: `makecontourplots.C`

- at the requested CLs level, typically 95% CL, CLs<0.05



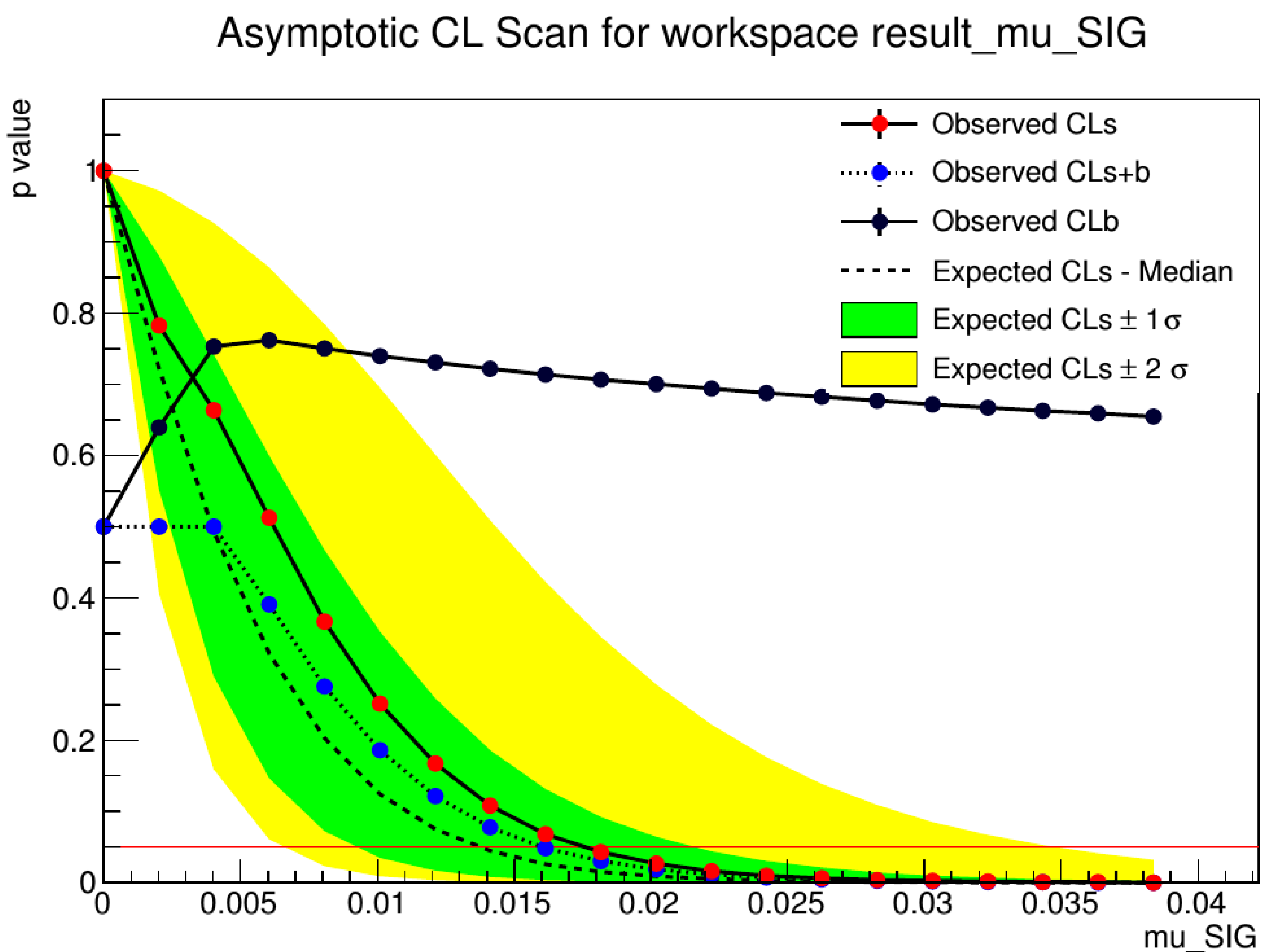
Signal strength upper limit

- Once you have unblinded your SR, one can set upper limits on specific signal models using the exclusion fit (aka model-dependent fit setup)
- As simple in HistFitter as calling:

```
HistFitter.py -l analysis/tutorial/MyUserAnalysis.py
```

  - Technicalities similar to '-p'

- Hypothesis test ***inversion***:
  - find the value of mu_SIG for which CLs below 0.05 (or other required value)
    - instead of calculating the p-value for the specific signal
  - run the hypothesis test for increasing values of signal strength mu_SIG
    - scan range determined automatically
    - upper limit on cross section = nominal cross section × upper limit on signal strength (grey numbers in contour plots, run for each signal grid point)



Model-independent upper limit

- Calculate the upper limit on the number of BSM physics events that we exclude in our SR
  - Typically used by theorists to check their favorite BSM model, that we have not looked at
- Requires the model-independent fit setup - aka discovery fit
  - ‘dummy signal’ = exactly one event in signal region (none in CRs)
  - upper limit on this ‘dummy signal’ = upper limit on BSM number of events
- Use the **UpperLimitTable.py** script:

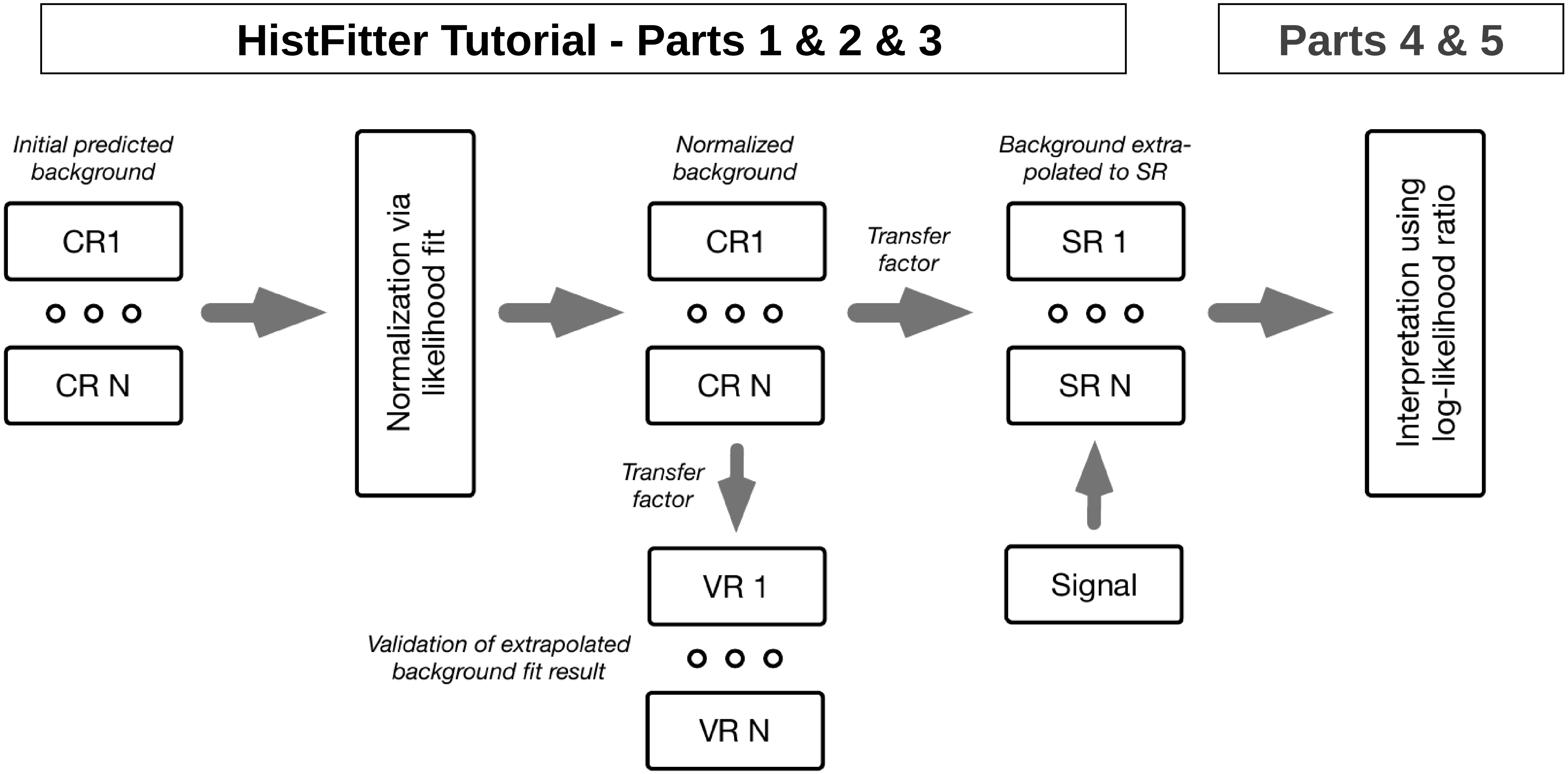
```
UpperLimitTable.py -c SS -w
results/MyUpperLimitAnalysis_SS/SPlusB_combined_NormalMeasurement_model.root -
l 4.713 -n 1000
```

- Results in LaTeX table:

Signal channel	$\langle\epsilon\sigma\rangle^{95}_{\mathrm{obs}}$[fb]	S^{95}_{obs}	S^{95}_{exp}	CL_B	$p(s=0)$
SS	1.73	8.2	$6.1^{+2.3}_{-1.3}$	0.80	0.21

  - ⟨σvis⟩95_obs : 95% CL upper limits on the visible cross section obs
  - S95_obs :95% CL upper limits on the number of signal events obs
  - S95_exp : 95% CL upper limit on the number of signal events, given the expected number (and ±1σ excursions on the expectation) of background events
  - CLB: the confidence level observed for the background-only hypothesis
  - p(s = 0): discovery p-value - the probability, capped at 0.5, that a background-only experiment is more signal-like than the observed number of events in a signal region

HistFitter - tutorial



HistFitter tutorial start up

- A public version is available on the HistFitter webpage:
  http://histfitter.web.cern.ch/histfitter/Software/Install/index.html

  We use HistFitter-2.0.tar.gz for this tutorial.

- **Installation instructions:**
  - Untar the HistFitter package
  - Setup ROOT (if not already done)
  - Go the HistFitter directory cd HistFitter-2.0
  - Run the HistFitter setup script source setup.sh
  - Go to the src/ directory and compile the C++ side of HistFitter cd src && make
  - Go back to the main HistFitter directory

  - **Input data here**:
    - Link the input data to your HistFitter directory as follows:
    - ln –s /project/etp3/jlorenz/shape_fit/samples/ samples